UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE INFORMÁTICA

ANÁLISIS Y EVALUACIÓN DE ALGORITMOS DE DETECCIÓN DE MOVIMIENTO Y SU APLICACIÓN A SISTEMAS DE SEGUIMIENTO EN VÍDEO

PROYECTO FIN DE CARRERA
INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

AUTOR: **LUIS RODRÍGUEZ LÓPEZ**
TUTOR: **MIGUEL ÁNGEL PATRICIO GUISADO**

JUNIO, 2009

ÍNDICE

1. INTRODUCCIÓN

1.1. AMBIENTACIÓN

En aplicaciones de visión de ordenadores, tales como videos de vigilancia y análisis de movimiento humano, los objetos de interés son normalmente los objetos de foreground (objetos que aparecen en un primer plano) en movimiento en una secuencia de imágenes. Una manera efectiva de extraer el objeto de foreground es suprimir los puntos de background (el fondo de la imagen) en los frames de la imagen. Para lograr esto, es deseable un modelo preciso y adaptativo de background.

El background normalmente contiene objetos no vivos que permanecen pasivos en la escena. Estos objetos pueden ser estacionarios, tales como paredes, puertas o el mobiliario de una habitación, o no estacionarios, como arbustos o escaleras en movimiento. La apariencia de los objetos de background frecuentemente experimenta cambios a lo largo del tiempo, por ejemplo, los cambios en la claridad causados por condiciones del clima cambiantes o el apagado/encendido de luces. La imagen de background se puede describir, por lo tanto, como formada por píxeles estáticos y dinámicos. Los píxeles estáticos pertenecen a los objetos estacionarios, y los píxeles dinámicos están asociados con objetos no estacionarios. La parte de background estático puede llegar a ser dinámica con el paso del tiempo, por ejemplo, mediante el encendido de una pantalla del ordenador. Y por otro lado, un píxel de background dinámico puede también pasar a ser estático, tal como sucede con un píxel perteneciente a un arbusto cuando el viento para. Para describir una escena de background general, un modelo de background debe ser capaz de

1) representar la apariencia de un píxel de background estático;
2) representar la apariencia de un píxel de background dinámico;
3) evolucionar a cambios de background graduales;
4) evolucionar a cambios de background repentinos.

Consideramos el problema de videos de vigilancia y seguimiento. Un sistema robusto no dependerá de la colocación cuidadosa de las cámaras. Además deberá ser robusto para cualquier objeto que esté en su campo de visión o cualquier efecto de luz que ocurra. Deberá ser capaz de tratar con movimientos en áreas desordenadas, objetos superponiéndose en el campo visual, sombras, cambios de iluminación, efectos de elementos en movimiento en la escena, objetos de lento movimiento, y objetos introducidos

o eliminados de la escena. Enfoques tradicionales basados en métodos de background generalmente fallaban en estas situaciones generales. Por tanto, la meta es crear un sistema rastreador robusto y adaptativo que sea bastante flexible para manejar variaciones en la iluminación, desorden de la escena en movimiento, múltiples objetos moviéndose y otros cambios arbitrarios para la escena observada.

1.2. MOTIVACIÓN

Se dispone de una aplicación en Microsoft Visual Studio 2005, donde se encuentran implementados los dos algoritmos "Foreground Detection based on background modeling and Bayes classification" (de aquí en adelante algoritmo FGD) y "Mixture of Gaussians" (de aquí en adelante algoritmo MOG), que son objeto de nuestro estudio, en C++. También se tuvo que instalar la librería OpenCV (Open Source Computer Vision Library), librería de tratamiento de imágenes, destinada principalmente a aplicaciones de visión por computador en tiempo real.

Con todo lo anterior, nuestra labor principal será experimentar, haciendo uso de dicha aplicación, sobre un conjunto de vídeos grabados en diferentes entornos y bajo diferentes condiciones climatológicas. Para ello, habrá que especificar en nuestra aplicación la ruta donde se encuentra el vídeo sobre el que se quiera realizar la experimentación. Igualmente especificaremos el algoritmo con el que se desea realizar el análisis, así como los valores para los parámetros propios del algoritmo que se haya escogido.

El objetivo de dicha experimentación será poder llegar a la conclusión de cuál de los dos algoritmos es el más adecuado, y con qué valores para sus parámetros, en función del vídeo (entorno) en el que realicemos el análisis.

1.3. OBJETIVO

Con la realización de este Proyecto Fin de Carrera se pretende cumplir con una serie de objetivos marcados previamente. Dichos objetivos pueden resumirse en los siguientes puntos:

- Análisis teórico de los algoritmos FGD y MOG
- Análisis de la implementación de los algoritmos con OpenCV.
- Estudio de las herramientas utilizadas para evaluar los algoritmos.
- Validación de los algoritmos FGD y MOG con distintos Datasets (conjuntos de datos) públicos.

A lo largo de este documento se verá detalladamente cada uno de estos puntos.

1.4. MEDIOS Y PLANIFICACIÓN

A la hora de comenzar el Proyecto, contamos con una serie de medios o recursos con los que partiremos para realizar las sucesivas tareas. Como se dijo anteriormente, disponemos de una aplicación en Microsoft Visual Studio 2005, donde se encuentran implementados los dos algoritmos FGD y MOG en C++. También se tuvo que instalar la librería OpenCV (Open Source Computer Vision Library), librería de tratamiento de imágenes, destinada principalmente a aplicaciones de visión por computador en tiempo real. Por otro lado, también contamos con documentación relativa a los dos algoritmos. Y por último, disponemos de una herramienta, *VirtualDub*, utilizada, entre otras cosas, para la edición de vídeos, y que tendremos que utilizar para generar los vídeos a partir de todos los Datasets (conjuntos de frames o imágenes) con los que también contamos para realizar sobre ellos la experimentación haciendo uso de los dos algoritmos.

Con todos esos recursos disponibles, procedemos a planificar todas las tareas que van a tener que realizarse a lo largo de este Proyecto Fin de Carrera. Se trata de citar estas tareas y dar una estimación de su duración en función del trabajo que conlleve cada una de ellas.

- **Análisis teórico de los algoritmos**. La primera tarea va a consistir en analizar la documentación que tenemos sobre los dos algoritmos. Con esto se pretenderá llegar a tener una idea de cómo funcionan. Si fuera necesario, se hará uso de documentación adicional que se pueda encontrar acerca de estos algoritmos. Se estima que un par de semanas pueden ser suficientes para conseguir tener una idea general sobre los algoritmos FGD y MOG.
- **Análisis de la implementación**. En esta segunda tarea se realizarán distintas actividades.
    - ⇒ *Edición de vídeos*. Haciendo uso de la herramienta *VirtualDub*, citada anteriormente, se procederá a generar vídeos en formato .avi partiendo de todos los Datasets de los que se dispone. Esta actividad es necesaria, ya que nuestra aplicación procesa vídeos en ese formato. Teniendo en cuenta la cantidad de Datasets que tenemos, se puede estimar que una semana puede ser suficiente para realizar esta actividad.
    - ⇒ *Implementación con OpenCV*. Se puede considerar una actividad auxiliar, que consiste, básicamente, en analizar la implementación con

OpenCV que hace referencia a los distintos módulos que tienen que ver con un sistema de videovigilancia, para así entender mejor nuestro código. En una semana se estima que puede llevarse a cabo esta actividad.

⇒ *Experimentación y conclusiones*. Posiblemente, la actividad que más tiempo va a necesitar. Se trata de procesar todos los vídeos haciendo uso de nuestra aplicación. Y para cada vídeo, tendremos que ver como se comporta el algoritmo FGD, y cómo lo hace el algoritmo MOG. Y no sólo eso, ya que, para cada algoritmo, tendremos que ir probando con diferentes valores para sus parámetros para ver como varía el comportamiento. Para hacerse una idea del tiempo que se puede tardar en realizar esta actividad, se puede decir que el tiempo medio para procesar un vídeo puede rondar los 30 minutos.

- **Medios de evaluación**. En esta tarea se estudiarán diferentes medios que existen para poder evaluar cuantitativamente el funcionamiento de los algoritmos. Se estudiará un servicio on-line de evaluación y diferentes métricas que evalúan los algoritmos de segmentación. Para llegar a cabo todo esto se prevee que serán necesarias un par de semanas.
- **Evaluación**. En esta última tarea, se escogerán un par de Datasets sobre los que realizar la evaluación. Para cada uno de ellos, generaremos los diferentes resultados que se obtienen en función del algoritmo y valores de los parámetros utilizados, y pasaremos a evaluar esos resultados haciendo uso de las métricas estudiadas en el punto anterior. Realizado esto, concluiremos exponiendo cuáles han sido el algoritmo y parámetros mejores para los dos Datasets seleccionados.

1.5. ESTRUCTURA DEL DOCUMENTO

Lo que resta de documento sigue una estructura directamente relacionada a los objetivos marcados en la sección 1.2., ya que, cada una de las secciones que se irán viendo corresponderá a cada uno de los objetivos establecidos. Estas secciones, además, se corresponden, en orden, con las tareas que se han ido realizando desde el comienzo del Proyecto hasta su finalización.

En la sección 2, Análisis teórico de los algoritmos FGD y MOG, se realizará una explicación sobre los dos algoritmos que son objeto de nuestro estudio, es decir, los algoritmos FGD y MOG. Consistirá en un estudio detallado, incluyendo aspectos técnicos, como son, por ejemplo, las fórmulas matemáticas en las que se fundamentan cada uno de los algoritmos.

En la sección 3, Análisis de la implementación, se detallará cómo se ha realizado todo el proceso de experimentación con la aplicación que disponemos. También se expondrán las diferencias más relevantes observadas entre los dos algoritmos una vez finalizada la experimentación en los distintos entornos de los que se dispone. De esta forma, se pretenderá concluir para qué tipo de vídeos será mejor un algoritmo que otro, y cuáles serán los parámetros más adecuados.

En la sección 4, Evaluación, se verán diferentes herramientas que existen para evaluar los algoritmos de segmentación. Por un lado, veremos una serie de métricas destinadas a evaluar los algoritmos. Y por otro lado, se analizará también el servicio de evaluación on-line que proporciona **PETS** (**P**erformance **E**valuation of **T**racking and **S**urveillance), donde se evalúan los diferentes algoritmos basándose en las distintas métricas explicadas, y los compara con otros con el mismo propósito.

En la sección 5, Validación / Experimentación, se explicará más detalladamente como se realiza la evaluación de los resultados que obtenemos de nuestra experimentación. En este proceso se detallará el Dasatet sobre el que se realiza la evaluación, la conversión de nuestros resultados a un formato adecuado para poder ser evaluado, y, finalmente, la evaluación de nuestros resultados.

El documento finalizará con las conclusiones, en la sección 6, que se han sacado del trabajo realizado. En esta sección se plantearán también futuros trabajos que se pueden abordar a raíz de todo el trabajo realizado en este Proyecto.

2. ANÁLISIS TEÓRICO DE LOS ALGORITMOS FGD Y MOG

En esta sección se realizará una explicación sobre los dos algoritmos que son objeto de nuestro estudio, es decir, los algoritmos FGD ("Foreground Detection based on background modeling and Bayes classification") y MOG ("Mixture of Gaussians"). Se intentará que la explicación sea lo más comprensible posible, pero al tratarse de un estudio detallado no se podrán obviar aspectos técnicos, como son las fórmulas matemáticas en las que se fundamentan cada uno de los algoritmos. La explicación de cada uno de ellos irá acompañada de un diagrama de flujo, tras la explicación, que sirva como resumen y ayude a comprenderlos en un simple vistazo.

De forma clara y concisa se puede decir que ambos métodos (algoritmos) se dirigen al problema de modelar el background para la detección de objetos de foreground en entornos complejos. A continuación veremos cada uno de ellos detenidamente.

2.1. Algoritmo FGD

Enfoque.-

El background es normalmente representado por características o rasgos de imagen en cada píxel. Las características extraídas de una secuencia de imágenes pueden ser clasificadas en tres tipos: características *espectrales*, *espaciales* y *temporales*. Las características *espectrales* están asociadas a escala de grises o información de color, las características *espaciales* están asociadas al gradiente o estructura local, y las características *temporales* están asociadas a cambios entre frames en el píxel. Las dos primeras son adecuadas para describir la apariencia de los píxeles de background estáticos, mientras que las terceras se usan para describir los píxeles de background dinámicos asociados con los objetos no estacionarios.

Este método o algoritmo propone un marco bayesiano que incorpora múltiples tipos de características para modelar backgrounds complejos. Los puntos importantes del método propuesto son los que siguen:

1) Se propone un marco bayesiano para incorporar características *espectrales*, *espaciales* y *temporales* en el modelado de background.
2) Se deriva una nueva fórmula de la regla de decisión de Bayes para la clasificación de background y foreground.
3) El background es representado usando estadísticas de las principales características asociadas con objetos de background estacionarios y no estacionarios.
4) Se propone un método nuevo para aprender y actualizar las características de background a cambios graduales y repentinos de background.
5) Se analiza la convergencia del proceso de aprendizaje y se deriva una formula para seleccionar un ratio de aprendizaje adecuado.
6) Se desarrolla un nuevo algoritmo en tiempo real para la detección objetos de foreground en entornos complejos.

Estos puntos de interés son desarrollados a continuación.

1) Clasificación de Bayes del background y foreground.-

Para objetos o regiones de background y foreground arbitrarios, su clasificación como background o foreground puede ser formulada bajo la teoría de decisión de Bayes.

Sea **s**=(x,y) la posición de un píxel de imagen, **I**(s,t) la imagen de entrada en el instante *t*, y **v** un vector característico *n*-dimensional extraído de la posición **s** en el instante *t* de la secuencia de imagen. Entonces, la probabilidad posterior del vector característico del background en **s** puede ser calculada mediante el uso de la regla de Bayes

$$P_s(b|\mathbf{v}) = P_s(\mathbf{v}|b)P_s(b) / P_s(\mathbf{v}) \quad (1)$$

donde *b* indica el background. $P_s(\mathbf{v}|b)$ es la probabilidad del vector característico **v** siendo observado como un background en **s**, $P_s(b)$ es la probabilidad anterior del píxel **s** perteneciendo al background, y $P_s(\mathbf{v})$ es la probabilidad anterior del vector característico **v** siendo observado en la posición **s**. De manera similar, la probabilidad posterior de que el vector característico **v** venga de un objeto de foreground en **s** es

$$P_s(f|\mathbf{v}) = P_s(\mathbf{v}|f)P_s(f) / P_s(\mathbf{v}) \quad (2)$$

donde *f* denota el foreground. Usando la regla de decisión de Bayes, un píxel **s** es clasificado perteneciente al background acorde a su vector característico **v** observado en el instante *t* si

$$P_s(b|\mathbf{v}) > P_s(f|\mathbf{v}) \quad (3)$$

De otra manera, es clasificado como perteneciente al foreground. Hay que tener en cuenta que un vector característico observado en un píxel de imagen viene de objetos de background o de objetos de foreground, esto supone:

$$P_s(\mathbf{v}) = P_s(\mathbf{v}|b)P_s(b) + P_s(\mathbf{v}|f)P_s(f) \quad (4)$$

Sustituyendo (1) y (4) en (3), se llega que la regla de decisión de Bayes (3) llega a ser

$$2P_s(\mathbf{v}|b)P_s(b) > P_s(\mathbf{v}) \quad (5)$$

Usando (5), el píxel **s** con vector característico observado **v** en el instante *t* puede ser clasificado como un punto de background o como un punto de foreground, con tal de que la anterior y condicional probabilidades $P_s(b)$, $P_s(\mathbf{v})$, y $P_s(\mathbf{v}|b)$ sean conocidas en adelante.

2) Representación de la característica principal del background.-

Para aplicar (5) para la clasificación del background y foreground, las funciones de probabilidad $P_s(b)$, $P_s(\mathbf{v})$, y $P_s(\mathbf{v}|b)$ o bien son conocidas en adelante, o pueden ser estimadas adecuadamente. Para backgrounds complejos, las formas de estas funciones de probabilidad son desconocidas, por lo que tendrán que ser estimadas. Una manera de estimar estas funciones de probabilidad es usando el histograma de características. El problema que se encuentra es el alto coste para almacenamiento y cálculo (computación). Asumiendo que **v** es un vector de *n*-dimensión y cada uno de sus elementos es cuantizado a L valores, el histograma contendría L^n celdas. Por ejemplo, asumiendo que la resolución de color tiene 256 niveles, el histograma contendría 256^3 celdas. El método sería irreal en términos de requerimientos computacionales y de memoria.

Es razonable asumir que si las características seleccionadas representan el background eficazmente, la extensión de las características de background sería pequeña, lo que implica que la distribución de características de background estará altamente concentrada en una pequeña región del histograma. Además, las características de diversos objetos de foreground se esparcirían ampliamente en el espacio característico. Esto implica que, con una adecuada selección y cuantización de características, sería posible describir aproximadamente el background usando solamente un número pequeño de vectores característicos. Una estructura de información concisa para implementar tal representación de background es creada como se explica a continuación.

Sean $\mathbf{v}_i$ los vectores característicos cuantizados clasificados en orden descendiente con respecto a $P_s(\mathbf{v}_i|b)$ por cada píxel **s**. Entonces, para una selección adecuada de características, habría un pequeño entero $N(\mathbf{v})$, un valor de porcentaje alto M_1, y un valor

de porcentaje bajo M_2 (por ejemplo, M_1 = 80% ~ 90% y M_2 = 10% ~ 20%) tales que el background estaría bien aproximado por

$$\sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i|b) > M_1 \quad y \quad \sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i|f) < M_2 \qquad (6)$$

El valor de N(**v**) y la existencia de M_1 y M_2 dependen de la selección y cuantización de los vectores característicos. Los N(**v**) *vectores característicos* son definidos como las características principales del background en el píxel **s**.

Para aprender y actualizar las probabilidades anterior y condicional para los vectores característicos principales, se establece una tabla de estadísticas para las características principales posibles para cada tipo de característica en **s**. La tabla es denotada como

$$T_v(\mathbf{s}) = \{ \begin{matrix} p^t_{\mathbf{v}}(b) \\ \{S^t_{\mathbf{v}}(i)\}, \ i = 1,\ldots,M(\mathbf{v}) \end{matrix} \qquad (7)$$

donde $p^t_{\mathbf{v}}(b)$ es el $P_s(b)$ aprendido basado en la observación de las características **v** y $S^t_{\mathbf{v}}(i)$ registra las estadísticas de los M(**v**) vectores característicos más frecuentes (M(**v**) > N(**v**)) en el píxel **s**. Cada $S^t_{\mathbf{v}}(i)$ contiene tres componentes

$$S^t_{\mathbf{v}}(i) = \{ \begin{matrix} p^t_{\mathbf{v}i} = P_s(\mathbf{v}_i) \\ p^t_{\mathbf{v}i|b} = P_s(\mathbf{v}_i|b) \\ \mathbf{v}_i = (v_{i1},\ldots v_{iD(\mathbf{v})})^T \end{matrix} \qquad (8)$$

donde D(**v**) es la dimensión del vector característico **v**. Los $S^t_{\mathbf{v}}(i)$ en la tabla $T_v(\mathbf{s})$ son clasificados en orden descendente con respecto al valor $p^t_{\mathbf{v}i}$. Los primeros elementos N(**v**) de la tabla $T_v(\mathbf{s})$, junto con $p^t_{\mathbf{v}}(b)$, son usados en (5) para la clasificación de background y foreground. Los elementos desde N(**v**) hasta M(**v**) en la lista son usados como un buffer para aprender las nuevas características importantes a través de la actualización del background. Los valores de N(**v**) y M(**v**) son seleccionados empíricamente. Para las características estables del background, valores pequeños son bastante buenos. Mientras que para las características con variaciones, se requieren ligeramente grandes valores.

3) Selección de característica.-

La siguiente cuestión esencial para la representación de la característica principal es la selección de característica. Las características importantes de distintos objetos de background son diferentes, es decir, no es lo mismo que se trate de un objeto de background estacionario que de un objeto de background dinámico. Para lograr la representación eficaz y precisa de los píxeles de background con características

principales, el empleo de tipos adecuados de características es importante. Tres tipos de características, las características *espectral*, *espacial* y *temporal*, son usadas para el modelado de background.

a) *Características para Píxeles de Background Estáticos*: Para un píxel perteneciente a un objeto de background estacionario, las características más importantes y estables son su color y estructura local (gradiente). Por lo tanto, en este caso, se usan dos tablas para aprender las características principales. Estas son $T_c(\mathbf{s})$ y $T_e(\mathbf{s})$ con $\mathbf{c} = [R,G,B]^T$ y $\mathbf{e} = [g_x,g_y]^T$ representando los vectores de color y gradiente, respectivamente. Debido a que el gradiente es menos sensitivo a cambios de iluminación, los dos tipos de vectores característicos pueden ser integrados bajo el marco de Bayes como se muestra a continuación.

Sea $\mathbf{v} = [\mathbf{c}^T,\mathbf{e}^T]^T$ y se asume que la **c** y **e** son independientes, la regla de decisión de Bayes (5) llega a ser

$$2P_s(\mathbf{c}|b)P_s(\mathbf{e}|b)P_s(b) > P_s(\mathbf{c})P_s(\mathbf{e}) \quad (9)$$

Para las características de los píxeles de background estáticos, la medida de cuantización sería menos sensitiva a cambios de iluminación. Aquí, una medida de distancia normalizada basada en el producto interno de dos vectores es empleada por ambos vectores de color y gradiente. La medida de distancia es

$$d(\mathbf{v}_1,\mathbf{v}_2) = 1 - 2(\mathbf{v}_1,\mathbf{v}_2) / ||\mathbf{v}_1||^2 + ||\mathbf{v}_2||^2 \quad (10)$$

donde **v** puede ser **c** o **e**, respectivamente. Si $d(\mathbf{v}_1,\mathbf{v}_2)$ es menor que un pequeño valor δ, $\mathbf{v}_1$ y $\mathbf{v}_2$ están casados entre si. El vector de color **c** es obtenido directamente de las imágenes de entrada con 256 niveles de resolución para cada componente, mientras que el vector de gradiente **e** es obtenido mediante la aplicación de operador Sobel para las correspondientes imágenes de entrada de escala de grises con 256 niveles de resolución. Con δ = 0.005, N(**v**) = 15 se encuentra precisión suficiente para aprender las características principales para los píxeles de background estáticos.

b) *Características para Píxeles de Background Dinámicos*: Para píxeles de background dinámicos asociados con objetos no estacionarios, las co-ocurrencias de color son usadas como sus características dinámicas. Esto es porque la co-ocurrencia de color entre frames consecutivos es considerada adecuada para describir las características dinámicas asociadas con objetos de background no estacionarios. Dando un cambio entre-frames del color $\mathbf{c}_{t-1} = [R_{t-1},G_{t-1},B_{t-1}]^T$ a $\mathbf{c}_t = [R_t,G_t,B_t]^T$ en el instante de tiempo *t* y el píxel **s** ($\mathbf{c}_t \neq \mathbf{c}_{t-1}$), el vector característico de co-ocurrencia de color es definido como $\mathbf{v} = \mathbf{cc} = [R_{t-1},G_{t-1},B_{t-1},R_t,G_t,B_t]^T$. De manera similar, una tabla de estadísticas para co-ocurrencia de color $T_{cc}(s)$

es mantenida en cada píxel. Sea $I(\mathbf{s},t) = [I_R(\mathbf{s},t),I_G(\mathbf{s},t),I_B(\mathbf{s},t)]^T$ la imagen de color entrante; el vector de co-ocurrencia de color **cc** es generado por medio de la cuantización de los componentes de color a baja resolución. Por ejemplo, mediante la cuantización de la resolución de color a 32 niveles para cada componente y seleccionando N(**cc**) = 50, se obtendría una representación característica principal buena para un píxel de background dinámico. Comparado con el espacio característico de co-ocurrencia de color de 32^6 celdas, N(**cc**) = 50 implica que con un número muy pequeño de vectores característicos, las características principales son capaces de modelar los píxeles de background dinámicos.

4) Aprendizaje y actualización de las estadísticas para las características principales.-

Dado que el background experimenta cambios tanto graduales como repentinos, se proponen dos estrategias para aprender y actualizar las estadísticas para las características principales.

a) *Para cambios de background graduales.-*

En cada instante de tiempo, si el píxel **s** es identificado como un punto estático, las características de color **c** y gradiente **e** son usadas para su clasificación como foreground o background. De otra manera, se usa la característica de co-ocurrencia de color **cc**. Asumamos que el vector característico **v** es usado para clasificar el píxel **s** en el tiempo *t* basándose en características principales aprendidas previamente. Entonces las estadísticas de los vectores característicos correspondientes en la tabla $T_{\mathbf{v}}(\mathbf{s})$ (**v** = **c** y **e**, o **cc**) son gradualmente actualizadas en cada instante de tiempo por

$$
\begin{aligned}
p^{t+1}{}_{v}(b) &= (1 - \alpha)p^{t}{}_{v}(b) + \alpha L^{t}{}_{b} \\
p^{t+1}{}_{vi} &= (1 - \alpha)p^{t}{}_{vi} + \alpha L^{t}{}_{vi} \\
p^{t+1}{}_{vi|b} &= (1 - \alpha)p^{t}{}_{vi|b} + \alpha(L^{t}{}_{b}L^{t}{}_{vi}) \qquad (11)
\end{aligned}
$$

donde el ratio de aprendizaje α es un número positivo pequeño y $i = 1,\ldots,M(\mathbf{v})$. En (11), $L^t_b = 1$ significa que **s** es clasificado como un punto de background en el tiempo *t* en la segmentación final, de otra manera, $L^t_b = 0$. De manera similar, $L^t_{\mathbf{v}i} = 1$ significa que el vector *i*th de la tabla $T_v(s)$ concuerda con el vector característico **v**, y de otra manera $L^t_{\mathbf{v}i} = 0$.

La operación de actualización anterior manifiesta lo siguiente. Si el píxel **s** es etiquetado como un punto de background en el tiempo *t*, $p^{t+1}{}_{\mathbf{v}}(b)$ es incrementado ligeramente respecto a $p^t_{\mathbf{v}}(b)$ debido a $L^t_b = 1$. Además, las probabilidades para el vector característico casado son también incrementadas debido a $L^t_{\mathbf{v}i} = 1$. Sin embargo, si $L^t_{\mathbf{v}i} = 0$,

entonces las estadísticas para los vectores característicos que no casan son decrementadas ligeramente. Si no hay concordancia entre el vector característico **v** y los vectores en la tabla $T_{\mathbf{v}}(\mathbf{s})$, el vector M(**v**)th en la tabla es reemplazado por un vector característico nuevo

$$p^{t+1}_{\mathbf{v}M(\mathbf{v})} = \alpha, \quad p^{t+1}_{\mathbf{v}M(\mathbf{v})|b} = \alpha, \quad \mathbf{v}_{M(\mathbf{v})} = \mathbf{v} \tag{12}$$

Si el píxel **s** es etiquetado como un punto de foreground en el tiempo *t*, $p^{t+1}_{\mathbf{v}}(b)$ y $p^{t+1}_{\mathbf{v}i|b}$ son decrementados ligeramente con $L^t_b = 0$. Sin embargo, el vector casado en la tabla $p^{t+1}_{\mathbf{v}i}$ es incrementado ligeramente.

Los elementos actualizados en la tabla $T_{\mathbf{v}}(\mathbf{s})$ son reclasificados (reordenados) en orden descendente con respecto a $p^{t+1}_{\mathbf{v}i}$, tal que la tabla mantendrían los vectores característicos M(**v**) mas frecuentes e importantes observados en el píxel **s**.

b) *Para cambios de background repentinos.-*

Acorde a (4), las estadísticas de las características principales satisfacen

$$\sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i) = P_s(b) \sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i|b) + P_s(f) \sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i|f) \tag{13}$$

Estas probabilidades son aprendidas gradualmente con operaciones descritas por (11) y (12) en cada píxel **s**. Cuando se produce un cambio de background repentino, la nueva apariencia del background pronto llega a ser dominante después del cambio. Con la operación respuesta (12), la operación de acumulación gradual (11) y reordenando en cada paso de tiempo, las características nuevas aprendidas serán gradualmente movidas a las primeras posiciones pequeñas en $T_{\mathbf{v}}(\mathbf{s})$. Después de un tiempo de duración, el término en la parte izquierda de (13) llega a ser grande (≈1) y el primer término en la parte derecha llega a ser muy pequeño debido a que las nuevas características de background son clasificadas como foreground. De (6) y (13), nueva apariencia de background en **s** puede ser encontrada si

$$P_s(f) \sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i|f) = \sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i) - P_s(b) \sum_{i=1}^{N(\mathbf{v})} P_s(\mathbf{v}_i|b) > M_1 \tag{14}$$

En (14), *b* denota el background anterior antes del cambio repentino y *f* denota la nueva apariencia del background después del cambio repentino. El factor $P_s(f)$ previene errores causados por un pequeño número de características de foreground. Usando la notación en (7) y (8), la condición (14) llega a ser

$$\sum_{i=1}^{N(\mathbf{v})} p^t_{\mathbf{v}i} - p^t_{\mathbf{v}}(b) \sum_{i=1}^{N(\mathbf{v})} p^t_{\mathbf{v}i|b} > M_1 \tag{15}$$

Una vez que la condición anterior es satisfecha, las estadísticas para el foreground son afinadas para ser la nueva apariencia de background. Acorde a (4), la operación de aprendizaje repentina es realizada como sigue:

$$p^{t+1}_{\mathbf{v}}(b) = 1 - p^{t}_{\mathbf{v}}(b)$$
$$p^{t+1}_{\mathbf{v}i} = p^{t}_{\mathbf{v}i}$$
$$p^{t+1}_{\mathbf{v}i|b} = (p^{t}_{\mathbf{v}i} - p^{t}_{\mathbf{v}}(b)p^{t}_{\mathbf{v}i|b}) / p^{t+1}_{\mathbf{v}}(b) \quad (16)$$

para $i = 1,\ldots,N(\mathbf{v})$.

5) Convergencia del proceso de aprendizaje y selección del ratio de aprendizaje.-

La convergencia del proceso de aprendizaje es analizada a continuación, y se deriva una fórmula para seleccionar el ratio de aprendizaje adecuado.

a) Convergencia del proceso de aprendizaje.-

Si la representación de característica principal ha aproximado satisfactoriamente el background, entonces $\sum^{N(\mathbf{v})}_{i=1} p^{t}_{\mathbf{v}i|b} \approx 1$ sería satisfecha. Además, es deseable que $\sum^{N(\mathbf{v})}_{i=1} p^{t}_{\mathbf{v}i|b}$ converja a 1 con la evolución del proceso de aprendizaje. Se puede mostrar que la operación de aprendizaje (11) verdaderamente se encuentra tal como una condición.

Supongamos $\sum^{N(\mathbf{v})}_{i=1} p^{t}_{\mathbf{v}i|b} = 1$ en el tiempo *t*, y el vector *j*th en la tabla $T_{\mathbf{v}}(\mathbf{s})$ casa con el vector característico de entrada **v** que ha sido detectado como background en la segmentación final en el tiempo *t*. Entonces, de acuerdo a (11), se tiene

$$\sum^{N(\mathbf{v})}_{i=1} p^{t+1}_{\mathbf{v}i|b} = (1 - \alpha) \sum^{N(\mathbf{v})}_{i=1} p^{t}_{\mathbf{v}i|b} + \alpha(L^{t}_{b} L^{t}_{\mathbf{v}j}) = 1 - \alpha + \alpha = 1 \quad (17)$$

que implica que la suma de las probabilidades condicionales de las características principales siendo background permanecerá igual o cercana a 1 durante la evolución del proceso de aprendizaje.

Supongamos $\sum^{N(\mathbf{v})}_{i=1} p^{t}_{\mathbf{v}i|b} \neq 1$ en el tiempo t debido a algunas razones tales como el alboroto de objetos de foreground o la operación de aprendizaje repentino, y el $\mathbf{v}_j$, de los primeros vectores $N(\mathbf{v})$ en $T_{\mathbf{v}}(\mathbf{s})$ casan con el vector característico de entrada **v**, luego tenemos

$$\sum^{N(\mathbf{v})}_{i=1} p^{t+1}_{\mathbf{v}i|b} = (1 - \alpha) \sum^{N(\mathbf{v})}_{i=1} p^{t}_{\mathbf{v}i|b} + \alpha(L^{t}_{b} L^{t}_{\mathbf{v}j}) \quad (18)$$

Si el píxel **s** es detectado como un punto de background en el tiempo *t*, ello conduce a

$$\sum_{i=1}^{N(\mathbf{v})} p^{t+1}_{\mathbf{v}i|b} - \sum_{i=1}^{N(\mathbf{v})} p^{t}_{\mathbf{v}i|b} = \alpha\left(1 - \sum_{i=1}^{N(\mathbf{v})} p^{t}_{\mathbf{v}i|b}\right) \quad (19)$$

Si $\sum_{i=1}^{N(\mathbf{v})} p^{t}_{\mathbf{v}i|b} < 1$, entonces $\sum_{i=1}^{N(\mathbf{v})} p^{t+1}_{\mathbf{v}i|b} > \sum_{i=1}^{N(\mathbf{v})} p^{t}_{\mathbf{v}i|b}$. En este caso, la suma de las probabilidades condicionales de las características principales siendo background incrementa ligeramente. Por otra parte, si $\sum_{i=1}^{N(\mathbf{v})} p^{t}_{\mathbf{v}i|b} < 1$, será $\sum_{i=1}^{N(\mathbf{v})} p^{t+1}_{\mathbf{v}i|b} > \sum_{i=1}^{N(\mathbf{v})} p^{t}_{\mathbf{v}i|b}$, y la suma de las probabilidades condicionales de las características principales siendo background decrementan ligeramente. De estos dos casos, puede ser concluido que la suma de las probabilidades condicionales de las características principales siendo background converge a 1 mientras que las características de background sean observadas frecuentemente.

b) Selección del ratio de aprendizaje.-

Para hacer el proceso de aprendizaje adaptable a los cambios de background graduales y no ser perturbado por el ruido y objetos de foreground, un valor pequeño sería seleccionado para α. Por otra parte, si α es demasiado pequeño, el sistema llegaría a ser demasiado lento para responder a los cambios de background repentinos. Aquí, se deriva una fórmula para seleccionar α acorde al tiempo requerido por el sistema para responder a los cambios de background repentinos.

Un cambio de background repentino ideal en el tiempo t_0 puede ser asumido para ser una función de paso. Suponiendo que las características antes de t_0 caen hacia los primeros K_1 vectores en la tabla $T_{\mathbf{v}}(\mathbf{s})$ ($K_1 < N(\mathbf{v})$), y las características después de t_0 caen hacia los siguientes K_2 elementos de $T_{\mathbf{v}}(\mathbf{s})$ ($K_2 < N(\mathbf{v})$). Entonces, las estadísticas en el tiempo t_0 pueden ser descritas como

$$\sum_{i=1}^{K1} p^{t0}_{\mathbf{v}i} \approx 1,\ \sum_{i=1}^{K1} p^{t0}_{\mathbf{v}i|b} \approx 1,\ p^{t0}_{\mathbf{v}}(b) \approx 1$$
$$\sum_{i=k1+1}^{K1+k2} p^{t0}_{\mathbf{v}i} \approx 0,\ \sum_{i=k1+1}^{K1+k2} p^{t0}_{\mathbf{v}i|b} \approx 0 \quad (20)$$

Debido a que la nueva apariencia de background en el píxel **s** después del tiempo t_0 es clasificado como foreground antes de la actualización repentina con (16), $p^{tk}_{\mathbf{v}}(b)$, $\sum_{i=1}^{K1} p^{tk}_{\mathbf{v}i}$ y $\sum_{i=1}^{K1} p^{tk}_{\mathbf{v}i|b}$ decrementan exponencialmente, mientras que $\sum_{i=k1+1}^{K1+k2} p^{tk}_{\mathbf{v}i}$ incrementa exponencialmente y será cambiado a las primeras K_2 posiciones en la tabla actualizada $T_{\mathbf{v}}(\mathbf{s})$ con clasificación a cada paso de tiempo. Inmediatamente la condición de (15) es encontrada en el tiempo t_n, el nuevo estado de background es aprendido. Para hacer la expresión más simple, asumamos que no hay operación de reordenación. Entonces la condición (15) llega a ser

$$\sum_{i=k1+1}^{K1+k2} p^{tn}_{\mathbf{v}i} - p^{tn}_{\mathbf{v}}(b) \sum_{i=k1+1}^{K1+k2} p^{tn}_{\mathbf{v}i|b} > M_1 \quad (21)$$

De (11) y (20), se sigue que en el tiempo t_n, las condiciones siguientes sostienen:

$$p^{tn}_{\mathbf{v}}(b) = (1-\alpha)^n \; p^{t0}_{\mathbf{v}}(b) = (1-\alpha)^n \tag{22}$$

$$\sum^{K1+k2}_{i=k1+1} p^{tn}_{\mathbf{v}i} = (1-\alpha)^n \sum^{K1+k2}_{i=k1+1} p^{t0}_{\mathbf{v}i} + \sum^{n-1}_{j=0}(1-\alpha)^j \, \alpha \approx 1-(1-\alpha)^n \tag{23}$$

$$\sum^{K1+k2}_{i=k1+1} p^{tn}_{\mathbf{v}i|b} = (1-\alpha)^n \sum^{K1+k2}_{i=k1+1} p^{t0}_{\mathbf{v}i|b} + 0 \approx 0 \tag{24}$$

Mediante sustitución (22)-(24) a (21) y ordenando de nuevo los términos, uno puede obtener

$$\alpha > 1-(1-M_1)^{1/n} \tag{25}$$

donde *n* es el número de frames requeridos para aprender la nueva apariencia de background. La ecuación (25) implica que si uno desea el nuevo sistema para aprender el nuevo estado de background en no más tarde que *n* frames, uno escogería α, de tal manera que (25) es satisfecha. Por ejemplo, si el sistema está para responder a un cambio repentino de background en 20 s con el ratio de frame siendo 20 fps y M_1 = 85%, α > 0.00473 sería satisfecho.

6) Detección de objeto de foreground: El algoritmo.-

Con la formulación bayesiana de la clasificación de background y foreground, además de la representación de background con características principales, se desarrolla un algoritmo para la detección de objeto de foreground. Éste consta de cuatro pasos: detección de cambio, clasificación de cambio, segmentación de objeto de foreground, y mantenimiento de background. En el primer paso, los píxeles de background que se mantienen inalterados en el frame actual son filtrados hacia fuera mediante el uso de background simple y diferenciación temporal. Los cambios detectados son separados en puntos estáticos y dinámicos acorde a cambios de interframe. En el segundo paso, los puntos de cambio estáticos y dinámicos detectados son también clasificados como background o foreground usando la regla de Bayes y las estadísticas de las características principales para background. Los puntos estáticos son clasificados basados en las estadísticas de los colores y gradientes principales, mientras que los puntos dinámicos son clasificados basados en esos de co-ocurrencias de color principales. En el tercer paso, los objetos de foreground son segmentados mediante la combinación de los resultados de clasificación de tanto los puntos estáticos como los puntos dinámicos. En el cuarto paso, los modelos de background son actualizados. Esto incluye actualizar las estadísticas de las características principales para el background además de una imagen de background de referencia. A continuación se describen los pasos brevemente:

I. Detección de cambio.-

En este paso, se usa diferenciación de imagen adaptativa simple para filtrar hacia fuera píxeles de background inalterados.

Sea $\mathbf{I}(\mathbf{s},t) = \{I_C(\mathbf{s},t)\}$ la imagen de entrada y $\mathbf{B}(\mathbf{s},t) = \{B_C(\mathbf{s},t)\}$ la imagen de background referenciada mantenida en tiempo t con C Є {R,G,B} denotando un componente de color. La diferencia de background es obtenida como sigue. Primero, se realizan la diferenciación y umbral de imagen para cada componente de color, donde el umbral es automáticamente generado usando el método de menor mediada de cuadrados (LMedS). La diferencia de background $F_{bd}(\mathbf{s},t)$ es entonces obtenida mediante fundición de los resultados de los tres componentes de color. De manera similar, la diferencia temporal (o interframe) $F_{td}(\mathbf{s},t)$ entre dos frames consecutivos $\mathbf{I}(\mathbf{s},t\text{-}1)$ y $\mathbf{I}(\mathbf{s},t)$ es obtenida. Si tanto $F_{bd}(\mathbf{s},t) = 0$ y $F_{td}(\mathbf{s},t) = 0$, el píxel **s** es clasificado como un punto de background inalterado. En general, mas del 50% de los píxeles serían filtrados hacia fuera en este paso.

II. Clasificación de cambio.-

Si $F_{td}(\mathbf{s},t) = 1$ es detectado en un píxel **s**, éste es clasificado como un punto dinámico, de otra manera, es clasificado como un punto estático. Un cambio que ocurre en un punto estático podría ser causado por cambios de iluminación, cambios de background repentinos, o un objeto de foreground inmóvil temporalmente. Un cambio detectado en un punto dinámico podría ser causado por un objeto de background o foreground moviéndose. Estos son también clasificados como background o foreground mediante el uso de la regla de decisión de Bayes y las estadísticas de las características principales correspondientes.

Sea $\mathbf{v}_t$ el vector característico de entrada en **s** y tiempo *t*. La probabilidades son estimadas como

$$P_s(b) = p^t_{\mathbf{v}}(b)$$

$$P_s(\mathbf{v}_t) = \sum_{\mathbf{v}j \in U(\mathbf{v}t)} p^t_{\mathbf{v}j}$$

$$P_s(\mathbf{v}t|b) = \sum_{\mathbf{v}j \in U(\mathbf{v}t)} p^t_{\mathbf{v}j|b} \qquad (26)$$

donde $U(\mathbf{v}_t)$ es una serie de vectores característicos compuestos de esos en $T_{\mathbf{v}}(\mathbf{s})$ que casan con el vector de entrada $\mathbf{v}_t$, por ejemplo

$$U(\mathbf{v}_t) = \{ \mathbf{v}_j \in T_{\mathbf{v}}(\mathbf{s}),\ d(\mathbf{v}_t,\mathbf{v}_j) \le \delta \text{ y } j \le N(\mathbf{v})\} \qquad (27)$$

Si no hay un vector característico principal en la tabla $T_v(\mathbf{s})$ que case con $\mathbf{v}_t$, tanto $P_s(\mathbf{v}_t)$ como $P_s(_{\mathbf{v}t}|b)$ son puestos a 0. Entonces, el punto de cambio es clasificado como background o foreground como sigue.

Clasificación de Punto Estático: Para un punto estático, las probabilidades para tanto las características de gradiente como de color son estimadas mediante (26) con **v** = **c** y **v** = **e**, respectivamente, donde la medida de distancia de vector $d(\mathbf{v}_1,\mathbf{v}_2)$ en (27) es calculada como (10). En este trabajo, las estadísticas de los dos tipos de características principales ($T_c(\mathbf{s})$ y $T_e(\mathbf{s})$) son aprendidos separadamente. En casos generales, sería $p^t_c(b) \approx p^t_e(b)$. La regla de decisión de Bayes (9) puede ser aplicada para clasificación de background y foreground.

Clasificación de Punto Dinámico: Para un punto dinámico en el tiempo *t*, el vector característico de co-ocurrencia de color $\mathbf{cc}_t$ es generado. Las probabilidades para $\mathbf{cc}_t$ son calculadas como (26), donde la distancia entre dos vectores característicos en (27) es calculada como

$$d(\mathbf{cc}_t, \mathbf{cc}_j) = \max_{k\in[1,6]}\{|\mathbf{cc}_{tk} - \mathbf{cc}_{jk}|\} \quad (28)$$

y δ = 2 es elegido. Finalmente, la regla de Bayes (5) es aplicada para clasificación de background y foreground. Generalmente, para los puntos de background dinámicos, sólo un pequeño porcentaje de ellos son clasificados erróneamente como cambios de foreground. Además, los restantes han llegado a ser puntos aislados, que pueden ser fácilmente eliminados por medio de una operación suave.

III. Segmentación de objeto de foreground.-

Se aplica un proceso posterior para segmentar los puntos de cambio restantes hacia regiones de foreground. Ésto es hecho primeramente mediante la aplicación de una operación morfológica (una pareja de *abrir* y *cerrar*) para suprimir los errores residuales. Luego las regiones de foreground son extraídas, los agujeros son rellenados y las regiones pequeñas son eliminadas. Además, una operación AND es aplicada a los segmentos resultantes en frames consecutivos para eliminar las regiones de foreground falsas detectadas mediante diferenciación temporal.

IV. Mantenimiento de background.-

Con la realimentación de la segmentación anterior, los modelos de background son actualizados. Primero, las estadísticas de las características principales son actualizadas como se describió anteriormente. Para los puntos estáticos, las tablas $T_{\mathbf{c}}(\mathbf{s})$ y $T_{\mathbf{e}}(\mathbf{s})$ son actualizadas. Para los puntos dinámicos, la tabla $T_{\mathbf{cc}}(\mathbf{s})$ es actualizada. Mientras tanto, una

imagen de background de referencia es también mantenida para hacer la conveniente diferencia de background.

Sea **s** un punto de background en el resultado de segmentación final en el tiempo *t*. Si es identificado como un punto de background inalterado en el paso de detección de cambio, la imagen de referencia de background en **s** es suavemente actualizada por

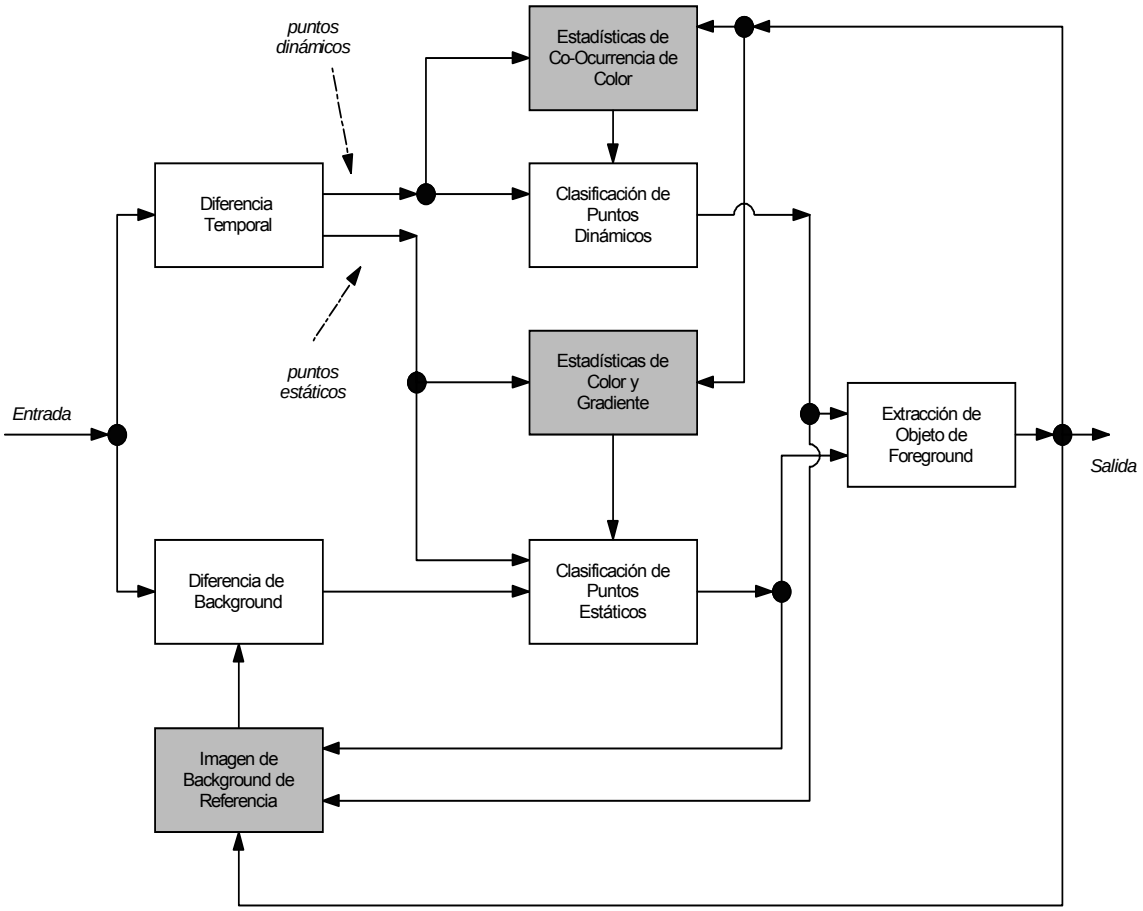
$$B_C(\mathbf{s},t+1) = (1 - \beta)B_C(\mathbf{s},t) + \beta I_C(\mathbf{s},t) \quad (29)$$

donde C Є {R,G,B} y β es un número positivo pequeño. Si **s** es clasificado como background en el paso de clasificación de cambio, la imagen de referencia de background en **s** es reemplazada por la nueva apariencia de background

$$B_C(\mathbf{s},t+1) = I_C(\mathbf{s},t), \text{ para } C \in \{R,G,B\} \quad (30)$$

Con (30), la imagen de background de referencia puede seguir los cambios de background dinámicos, por ejemplo, los cambios de color entre la rama de árbol y el cielo, además de cambios de background repentinos.

A continuación se muestra, a modo de resumen, un diagrama de flujo que ayuda a comprender más fácilmente el algoritmo FGD.



Los bloques blancos de izquierda a derecha corresponden a los primeros tres pasos, y los bloques con sombras grises corresponden al mantenimiento del background.

2.2. Algoritmo MOG

Enfoque.-

En este método o algoritmo se modelan los valores de un píxel particular como una mezcla de distribuciones gaussianas. Basándose en la persistencia y la concordancia de cada una de las gaussianas de la mezcla, se determina cuáles de ellas podrían corresponder a colores de background. Los valores de píxel que no encajan con distribuciones de background son considerados foreground hasta que haya una gaussiana que los incluya con evidencia suficiente y consistente para soportarlo.

Método.-

Antes de nada hay que saber que si un píxel resulta de una superficie concreta bajo una iluminación determinada, una distribución gaussiana simple sería suficiente para modelar el valor del píxel. Por otro lado, si solamente se producen cambios de iluminación durante horas, una distribución gaussiana simple y adaptativa sería suficiente. Pero en la práctica, son múltiples superficies las que aparecen a menudo en la vista de un píxel particular y las condiciones de iluminación cambian. Por lo tanto, en ese caso, son necesarias gaussianas múltiples adaptativas. Este método usa entonces una mezcla de gaussianas adaptativas para aproximar este proceso.

Cada cierto tiempo los parámetros de las gaussianas son actualizados. Las gaussianas son evaluadas usando una simple heurística para hacer hipótesis de cuáles son mas probables que sean parte del "proceso de background" (puntos 1 y 2). Los valores del píxel que no concuerdan con los píxeles del "background" son agrupados usando componentes relacionados (punto 3). Finalmente, los componentes relacionados son rastreados frame a frame usando un rastreador de hipótesis múltiple (punto 4). El proceso es detallado a continuación:

1) Modelo Mixto on-line.-

Se considera a los valores de un píxel particular durante un periodo de tiempo como un "proceso de píxel". El "proceso de píxel" es, por tanto, una serie de tiempo de valores del píxel.

Con un background estático e iluminación estática, ese valor sería relativamente constante. Desafortunadamente, como ya se ha dicho anteriormente, la mayoría de las secuencias de videos interesantes implican cambios de luz, cambios de escenas, y objetos moviéndose.

Hay que considerar, por tanto, distintos casos. Podrían producirse cambios de iluminación en una escena estática, por lo que sería necesario para la gaussiana rastrear esos cambios. También puede darse el caso de que un objeto estático sea añadido a la escena y no sea incorporado al background hasta que esté allí más tiempo que el objeto previo, por lo que los píxeles correspondientes serían considerados foreground durante largos periodos de tiempo. Esto conduciría a acumular errores en la estimación del foreground, resultando pobres comportamientos de rastreo. Estos factores proponen que las observaciones más recientes deberían ser más importantes a la hora de determinar las estimaciones de parámetros gaussianos.

Un aspecto adicional de variación ocurre si en la escena están presentes objetos moviéndose, ya que un objeto en movimiento coloreado relativamente de forma constante

se espera que produzca más variación que un objeto “estático”. Además, en general, habría más datos dando soporte a las distribuciones de background porque éstas son repetidas, mientras que los valores de píxel para diferentes objetos no son a menudo del mismo color.

Todos estos factores mencionados son los que sirven de guía a la hora de de modelar y actualizar el procedimiento. La historia reciente de cada píxel, {X1,…,Xt}, está modelada por una mezcla de K distribuciones gaussianas (K es determinado por la memoria disponible y el poder computacional. Actualmente su valor varía de 3 a 5).

Así pues, la distribución de los valores observados recientemente de cada píxel en la escena está caracterizada por una mezcla de gaussianas. Un nuevo valor de píxel estará, en general, representado por uno de los componentes principales del modelo mixto (es decir, por una de las distribuciones gaussianas de la mezcla) y será usado para actualizar el modelo.

Si el proceso de píxel fuera considerado un proceso estacionario, un método estándar para maximizar la probabilidad del dato observado es el *expectation maximation*. Desafortunadamente, como ya se dijo anteriormente, cada proceso de píxel varía durante el tiempo como el estado del mundo cambia, por lo que se usará un método aproximado que trata esencialmente cada nueva observación como una colección de muestra de tamaño 1 y usa reglas de aprendizaje estándar para integrar la nueva información.

Debido a que hay un modelo mixto para cada píxel en la imagen, implementar un algoritmo EM exacto sería costoso. En su lugar, se implementa una aproximación on-line de k medias. Cada nuevo valor del píxel, Xt, es cotejado con las distribuciones K gaussianas existentes, hasta que se encuentra coincidencia. Una coincidencia es definida como un valor de píxel dentro de una desviación estándar de 2.5 de una distribución. Este umbral puede ser perturbado con un pequeño efecto en la ejecución. Esto es realmente un umbral por píxel por distribución, y es extremadamente útil cuando diferentes regiones tienen diferente iluminación, porque los objetos que aparecen en regiones sombreadas no exhiben generalmente tanto ruido como los objetos en las regiones iluminadas. Un umbral uniforme a menudo resulta en objetos que desaparecen cuando éstos entran en regiones oscuras.

Si ninguna de las K distribuciones casan con el valor del píxel actual, la distribución menos probable es reemplazada por una distribución con el valor actual como su valor principal, una varianza alta inicialmente, y un peso de prioridad bajo.

Los pesos de prioridad de las K distribuciones en el instante t son ajustados.

Los parámetros μ y σ para distribuciones que no casan se quedan igual. Los parámetros de la distribución que casan con la nueva observación son actualizados.

Una de las ventajas importantes de este método es que si algo es permitido para formar parte del background, esto no destruye el modelo existente de background, ya que el color de background original permanece en la mezcla hasta que llega a ser el K^{th} más probable y un nuevo color es observado. Por lo tanto, si un objeto es estacionario lo suficientemente largo para formar parte del background y entonces se mueve, la distribución describe el background previo hasta que exista con el mismo μ y σ^2.

2) Estimación del modelo de background.

Como los parámetros del modelo mixto de cada píxel cambian, se quiere determinar cuál de las gaussianas de la mezcla son producidas más probablemente por los procesos de background. Interesa la distribución gaussiana que tienen la evidencia de más soporte y la menor variación.

Para comprender esta elección, se considera la acumulación de evidencias de soporte y la variación relativamente baja para las distribuciones de "background" cuando un objeto estático y persistente es visible. En contraste, cuando un objeto nuevo obstruye el objeto de background, este, en general, no casará con ninguna de las distribuciones existentes e implicará o bien la creación de una nueva distribución o en el incremento de la variación de una distribución existente. Además, la variación de un objeto en movimiento se espera que permanezca más larga que un píxel de background hasta que el objeto en movimiento pare. Para modelar esto, se necesita un método para decidir que porción del modelo mixto es la que mejor representa los procesos de background.

Para ello, primeramente, las gaussianas son ordenadas según su evidencia y variación. Esta ordenación del modelo es efectivamente una lista ordenada y sin límite, donde las distribuciones de background más probables permanecen en la cima y las distribuciones de background pasajeras menos probables gravitan hacia el fondo y son eventualmente sustituidas por nuevas distribuciones.

Entonces las primeras B distribuciones son elegidas como el modelo de background, donde

$$B = argmin_b(\sum_{k=1}^{b} \omega_k > T) \quad (1)$$

donde ω_k es una estimación del peso (qué porción de los datos es explicada por esta gaussiana) de la gaussiana k, y donde T es una medida de la porción mínima de la

información que debería ser explicada por el background. Esto toma la "mejor" distribución hasta una porción cierta, T, de la reciente información que ha sido explicada. Si se escoge un pequeño valor para T, el modelo de background es normalmente unimodal. Si este es el caso, usando solamente la distribución más probable salvaremos el proceso.

Si T es más alto, una distribución multimodal causada por un movimiento de background repetitivo (por ejemplo, las hojas de un árbol, una bandera en el viento, etc.) resultaría en que más de un color sería incluido en el modelo de background. Estos resultados permiten al background aceptar dos o más colores separados.

3) Componentes conectados.-

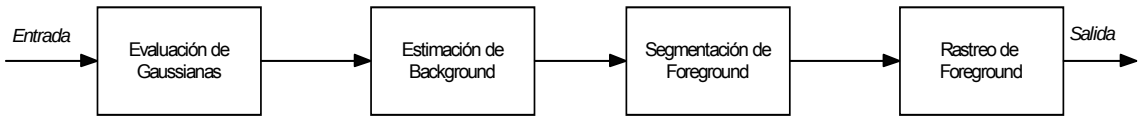
El método descrito anteriormente permite identificar píxeles de foreground en cada nuevo frame mientras se actualiza la descripción de cada proceso de píxel. Estos píxeles de foreground etiquetados pueden entonces ser segmentados en regiones por un algoritmo de componentes conectados en dos pasos.

Ya que este procedimiento es efectivo en determinados objetos en movimiento, las regiones en movimiento pueden ser caracterizadas no sólo por su posición, si no también por su tamaño, momentos (instantes, importancias), y otra información de forma. No sólo pueden estas características ser útiles para el posterior proceso y clasificación, si no que también pueden ayudar en el proceso de rastreo.

4) Rastreo de hipótesis múltiple.-

Estableciendo correspondencia de los componentes conectados entre frames se logra usar un algoritmo de rastreo de múltiples hipótesis predictivo directamente que incorpore tanto posición como tamaño. Debido a que este algoritmo no es esencial en el entendimiento del método, es preferible no profundizar en el funcionamiento del mismo debido a su complejidad.

A continuación se muestra, a modo de resumen, un diagrama de flujo que ayuda a comprender más fácilmente el algoritmo MOG.



3. ANÁLISIS DE LA IMPLEMENTACIÓN

En esta sección se detallará cómo se ha realizado todo el proceso de experimentación sobre nuestra aplicación haciendo uso de todos los vídeos disponibles.

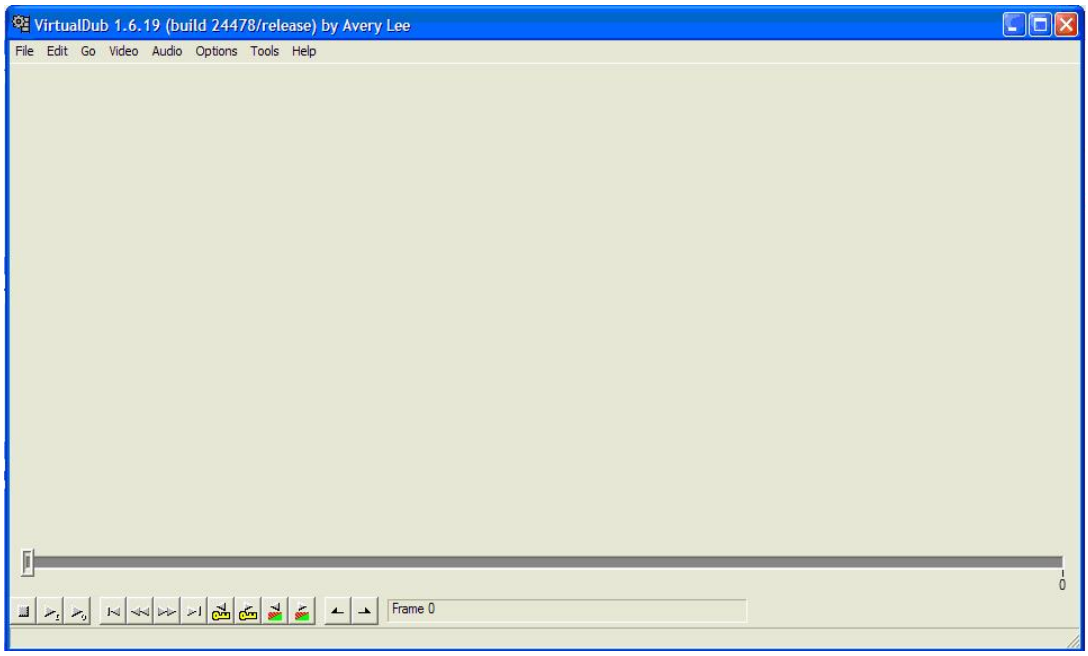
En primer lugar se verá cómo se generaron algunos de estos vídeos a partir del conjunto de imágenes o frames que los componen. A continuación, se explicará el funcionamiento de la aplicación que tenemos implementada con OpenCv. Con todo lo anterior realizado, pasaremos a detallar todo el proceso de experimentación, viendo para cada vídeo que algoritmo es el más adecuado y con qué parámetros, y haciendo una comparación de cómo se comportan los dos algoritmos en los puntos clave del vídeo. Por último, se expondrán, a modo de resumen, las diferencias más relevantes observadas entre los dos algoritmos una vez finalizada la experimentación en los distintos entornos de los que se dispone. De esta forma, se pretenderá concluir para qué tipo de vídeos será mejor un algoritmo que otro, y cuáles serán los parámetros más adecuados.

3.1. Edición de vídeos

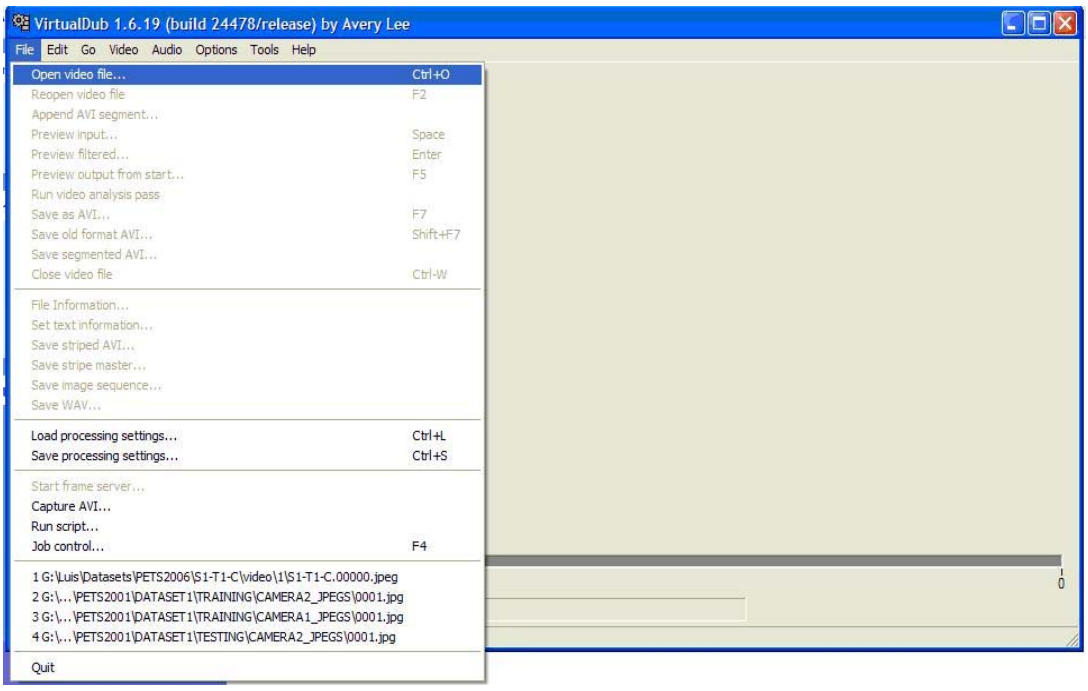
Como se dijo antes, la primera tarea realizada antes de comenzar con la experimentación, fue la recolección de un conjunto de vídeos sobre los que realizar las pruebas.

Ahora bien, algunos de esos vídeos fueron obtenidos ya en formato .AVI, pero otros han tenido que ser generados a partir de imágenes .JPG mediante un editor de vídeo. El programa utilizado para realizar esta labor ha sido el VirtualDub, cuyo funcionamiento se detalla a continuación:

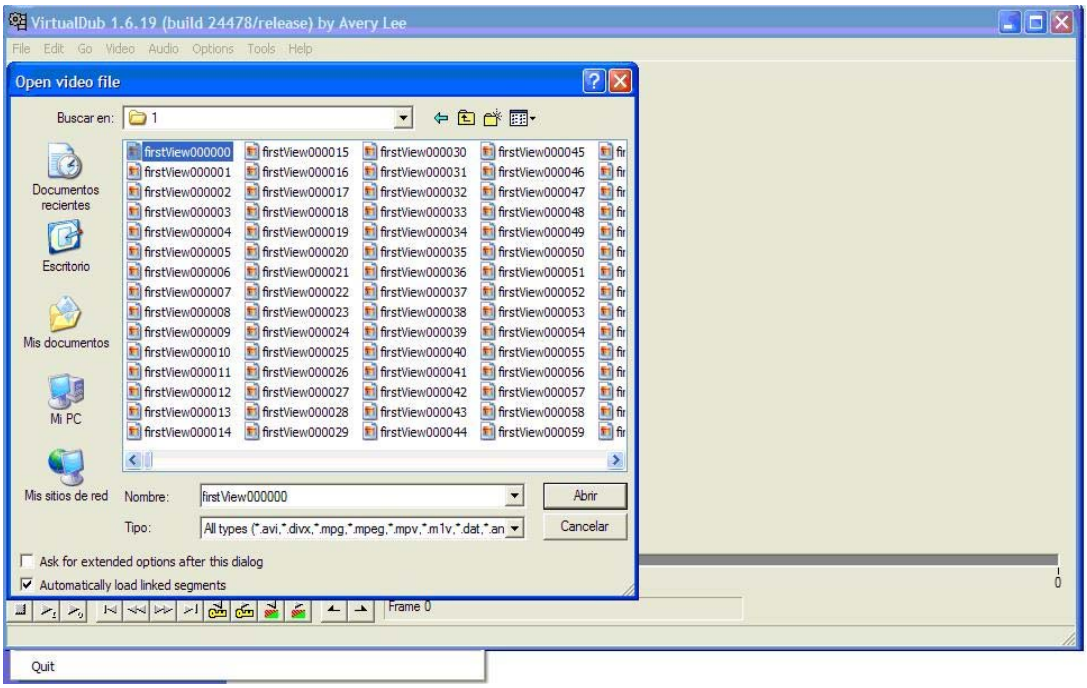
- Al ejecutar VirtualDub nos encontramos con la siguiente pantalla:



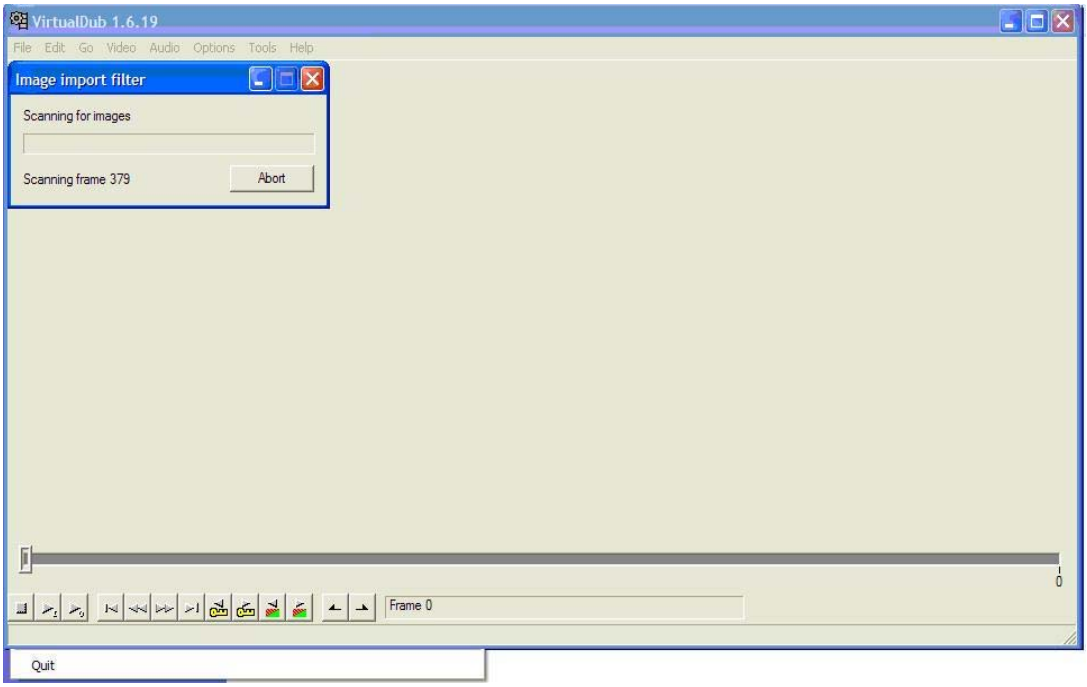
- Como se dijo, lo que se pretende es generar un vídeo a partir de un conjunto de imágenes ("frames"), por lo que primeramente será necesario indicar al programa donde se encuentran tales imágenes. Para ello, hay que ir al menú "*File*" y ahí seleccionar la opción "*Open video file…*".



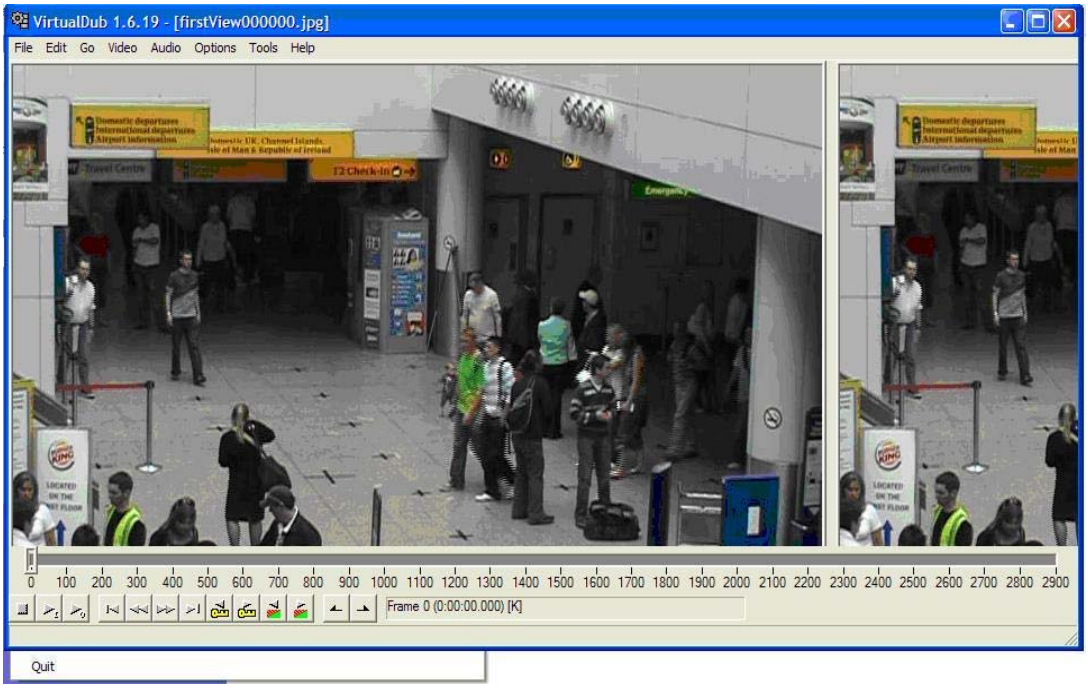
- Seleccionada dicha opción, tendremos que buscar el directorio donde se encuentren las imágenes y seleccionar la primera del conjunto.



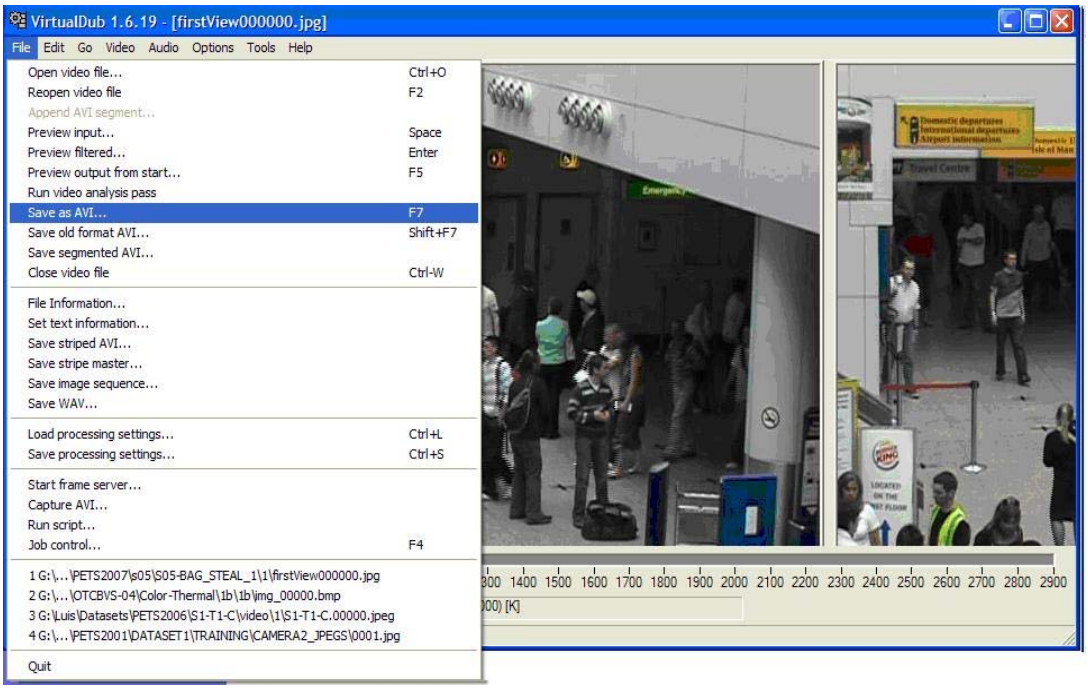
- Y pulsando en **Abrir**, la aplicación automáticamente comenzará a importar todas las imágenes del directorio en el orden en el que se encuentren. Como se puede apreciar en la captura de pantalla que se muestra a continuación, este proceso va mostrando el número de imágenes que lleva importadas la herramienta. También se puede observar que el proceso puede ser abortado en cualquier momento.



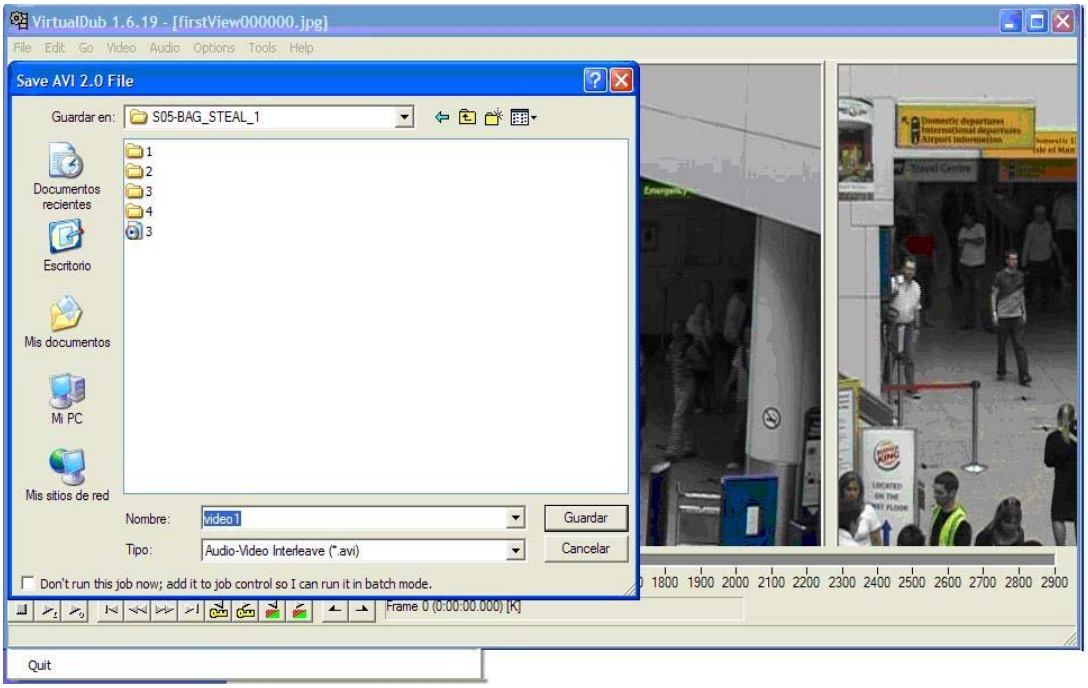
- Finalizado el paso anterior, nos encontramos en una situación como la que se muestra, donde las imágenes han sido importadas y se está a la espera de realizar la función que se desee.



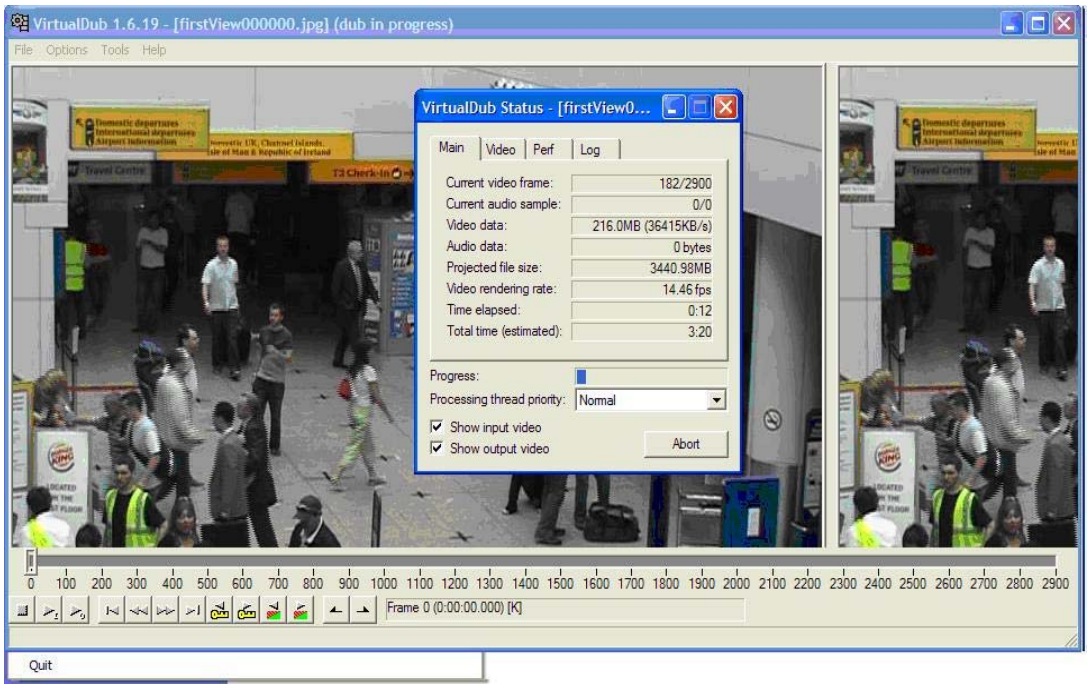
- A continuación, hay que indicar al programa la función a realizar, en nuestro caso, la de editar un vídeo. Para ello se accede nuevamente al menú "*File*" y se selecciona la opción "*Save as AVI…*".



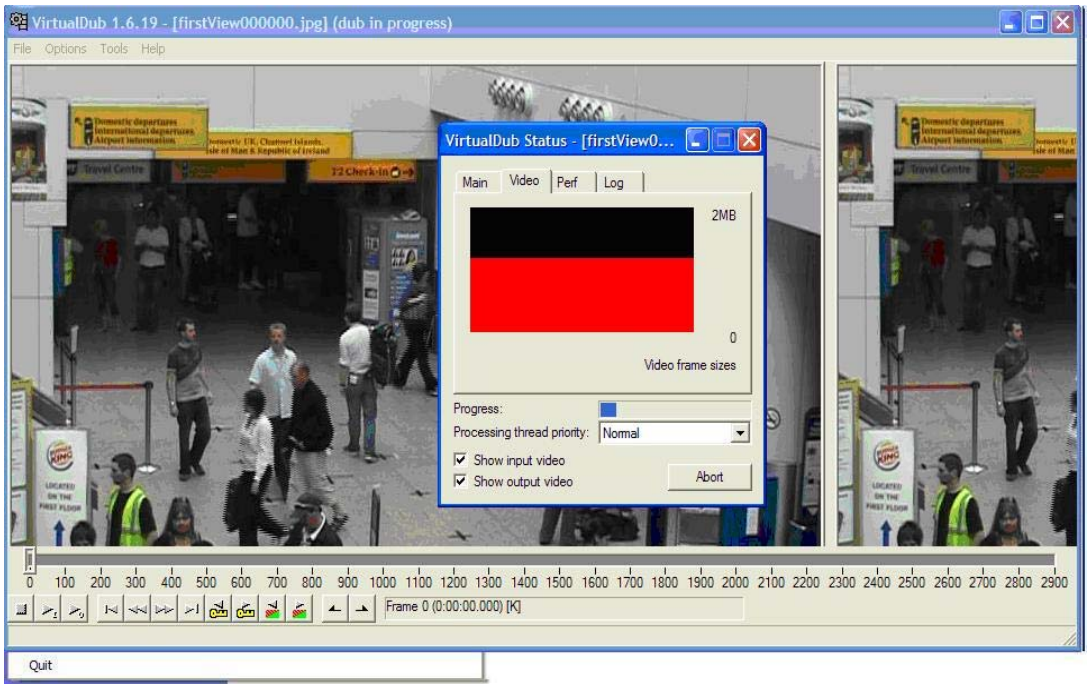
- Indicamos, seguidamente, dónde queremos ubicar el vídeo y con qué nombre.



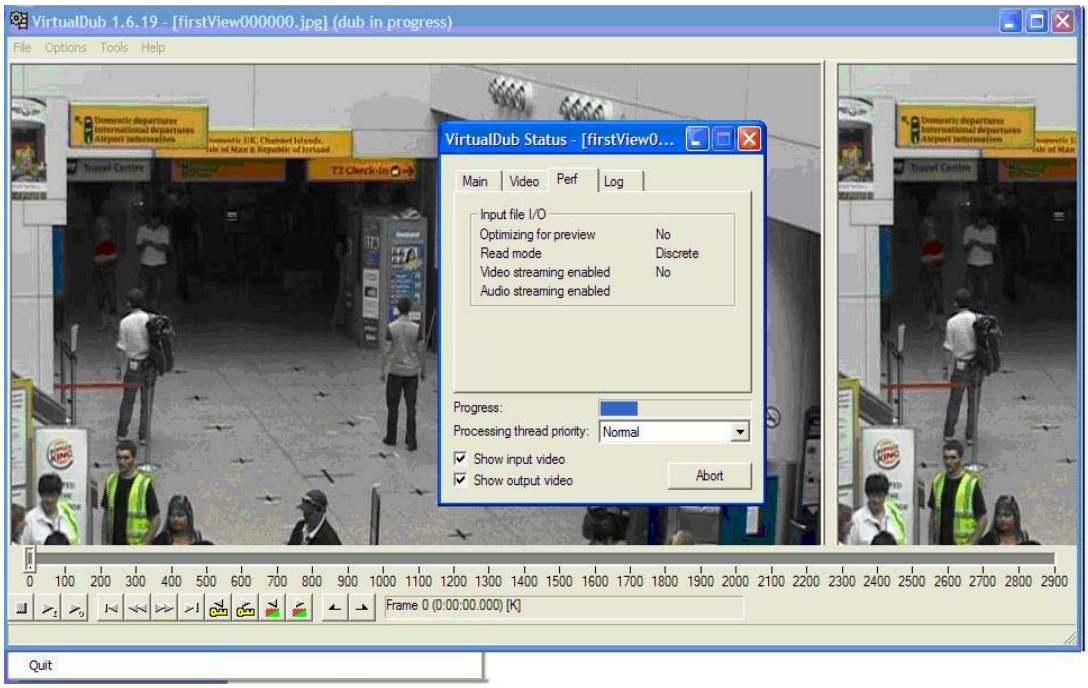
- Y pulsando en **Guardar**, la herramienta comienza a editar el vídeo encadenando cada una de las imágenes importadas anteriormente. Como se puede apreciar, se muestra una ventana donde se aporta información general del proceso en la parte inferior, como es: una barra que indica el progreso del mismo, la prioridad que se quiere asignar al procesamiento (Normal, Superior, Incluso Superior) y dos opciones que permiten, en el caso de estar marcadas, mostrar el vídeo de entrada (vídeo que se muestra en la parte izquierda) y el vídeo de salida (en la parte derecha), respectivamente. Además de esta información, se dispone de otra clasificada en cuatro pestañas:
    - *Main*: Donde se aporta la siguiente información:
        - Frame de video en el que se encuentra el proceso.
        - Información referente al audio, que en nuestro caso no es requerida.
        - El tamaño de vídeo que se lleva ocupado en cada instante, junto con la velocidad de transferencia.
        - El tamaño total del proyecto, es decir, del vídeo que se va a generar.
        - El número de frames que se van transfiriendo (encadenando) por segundo.
        - El tiempo que se lleva transcurrido del proceso.
        - El tiempo total estimado en llevar a cabo la tarea.



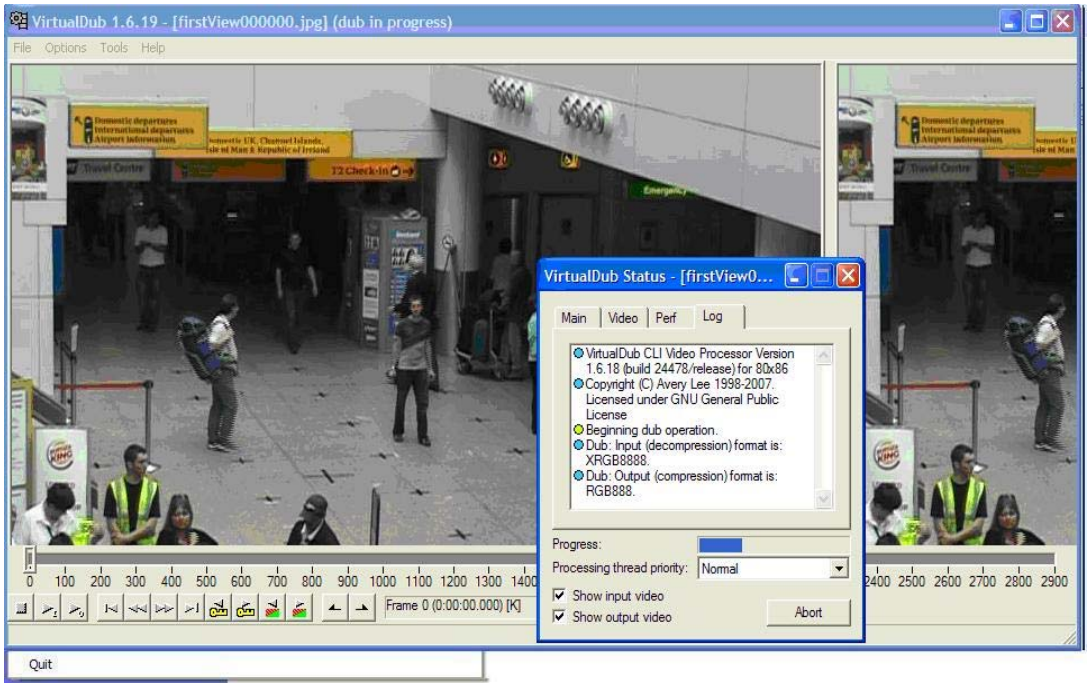
- *Video*: Se muestra el tamaño de los frames que componen el vídeo mediante una especie de histograma donde las barras rojas indican el tamaño de cada frame. En este caso puede observarse como las imágenes que componen el vídeo tienen un tamaño muy similar.



- *Perf*: Muestra información referente el fichero de entrada.



- *Log*: Muestra información referente al programa.



Este proceso, al igual que para el caso de la importación de imágenes, puede ser abortado en cualquier momento pulsando la opción **Abort**.

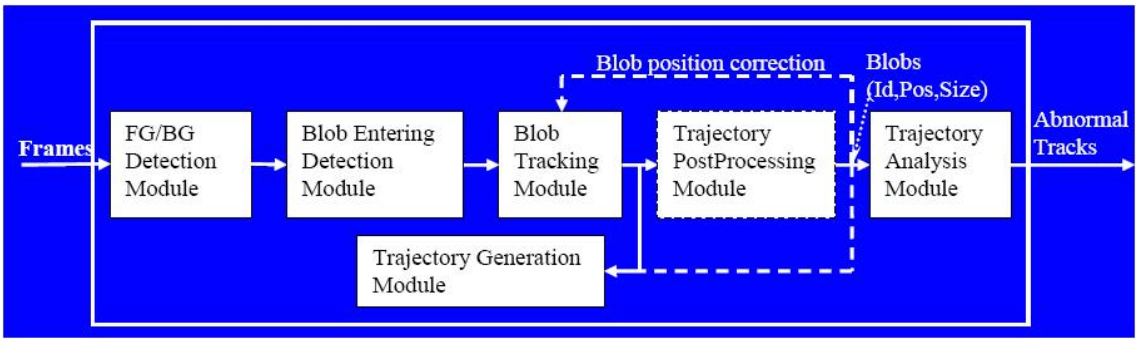
- La edición de vídeo termina, y tenemos nuestro vídeo .AVI en la ubicación deseada.

3.2. Implementación con OpenCV

OpenCV (Open Source Computer Vision) es una librería de funciones de programación orientadas principalmente a la visión por ordenador en tiempo real. Aunque son muchos los ejemplos de aplicaciones con esta librería, en este trabajo nos centraremos en el ámbito de los vídeos de vigilancia.

Tradicionalmente, la labor con vídeos de vigilancia ha sido intensa y, generalmente, no muy efectiva. Sin embargo, la videovigilancia con técnicas de visión por ordenador permite ahorrar en mano de obra, y proporciona una calidad constante de monitorización. La entrada a un sistema de videovigilancia es un vídeo proveniente de una cámara o de múltiples. El sistema analiza el contenido del vídeo mediante la separación del foreground del background, detectando y rastreando los objetos, y realizando un análisis a alto nivel. El análisis a alto nivel proporciona resultados tanto en un escenario que sea normal como en otro que no lo sea. El operador humano puede entonces centrarse en los escenarios anormales y no tiene que mirar en el vídeo intentando encontrar alguna anomalía.

A continuación se muestra un esquema general del sistema de videovigilancia (correspondiente a la tubería del sistema de videovigilancia)



donde

- Un modulo de "*Foreground/Background (FG/BG) Detection*" realiza la clasificación de FG/BG de cada píxel de imagen.
- Un modulo de "*Blob Entering Detection*" usa el resultado (máscara de FG) del módulo "FG/BG Detection" para detectar que un nuevo *blob object* (objeto que es detectado en el frame) entra en la escena.
- Un módulo de "*Blob Tracking*" es inicializado por el módulo de "Blob Entering Detection". Este módulo rastrea cada blob de la lista de blobs rastreados.

- Un módulo de "*Trajectory Generation*" recoge todas las posiciones de los blobs y guarda cada trayectoria de los blobs al disco duro cuando el movimiento del objeto no se presenta durante mucho tiempo (por ejemplo, cuando se pierde el rastreo).
- Un módulo de "*Trajectory Post Processing*" ejecuta una función suavizada sobre una trayectoria de blob. Este módulo es opcional y no necesita ser incluido en una tubería específica.
- Un módulo de "*Trajectory Analysis*" realiza un análisis de trayectorias de blob y detecta trayectorias anormales.

A continuación se detallará cada uno de los módulos anteriormente expuestos, incluyendo también la parte del código fuente de nuestra aplicación que hace referencia a cada uno de ellos.

Foreground/Background (FG/BG) Detection

Dada una secuencia de vídeo, la extracción del FG del BG es un paso importante en toda la tubería de videovigilancia. La estimación de FG es la primera etapa en la tubería. Su precisión (relativa a la precisión de los algoritmos) afecta a la precisión de las etapas posteriores. También podría afectar a la realización (relativa a aspectos computacionales, como por ejemplo la velocidad de computación) de las etapas posteriores.

La detección de FG es generalmente más temprana en los entornos de interior. Idealmente, lo que se quiere es una estimación de FG/BG para trabajar bien en ambos entornos de interior y exterior. El entorno de exterior es más complejo, ya que, como se ha dicho ya en este trabajo, pueden ocurrir movimientos en las ramas de un árbol, movimientos en superficies de agua, cierre y apertura de puertas de forma periódica, etc.

Para realizar esta labor, nosotros disponemos de los algoritmos **FGD** y **MOG**, que ya fueron explicados en el punto anterior (punto 2).

A continuación se muestra la parte del código fuente de nuestra aplicación asociada al módulo "*Foreground/Background (FG/BG) Detection*"

```
/* list of FG DETECTION modules */
static CvFGDetector* cvCreateFGDetector0(){return
cvCreateFGDetectorBase(CV_BG_MODEL_FGD, NULL);}
static CvFGDetector* cvCreateFGDetector0Simple(){return
cvCreateFGDetectorBase(CV_BG_MODEL_FGD_SIMPLE, NULL);}
static CvFGDetector* cvCreateFGDetector1(){return
cvCreateFGDetectorBase(CV_BG_MODEL_MOG, NULL);}
typedef struct DefModule_FGDetector
{
    CvFGDetector* (*create)();
    char* nickname;
```

```
    char* description;
} DefModule_FGDetector;
DefModule_FGDetector FGDetector_Modules[] =
{
    {cvCreateFGDetector0,"FGD","Foreground Object Detection from
Videos Containing Complex Background. ACM MM2003."},
    {cvCreateFGDetector0Simple,"FGDS","Simplyfied version of FGD"},
    {cvCreateFGDetector1,"MOG","Adaptive background mixture models
for real-time tracking. CVPR1999"},
    {NULL,NULL,NULL}
};
```

Blob Entering Detection

En esta etapa, los píxeles de foreground adyacentes son agrupados en blobs. Como se dijo anteriormente estos píxeles de foreground son resultado del módulo anterior.

Nosotros disponemos de dos alternativas para realizar esta labor, dos módulos, denominados **BD_CC** y **BD_Simple**.

El primero de ellos, **BD_CC**, está basado en un rastreador de componentes conectados, que hace lo siguiente:

1. Calcula componentes conectados de la máscara de FG obtenida por el módulo de estimación de FG/BG. Cada componente es considerado como un blob.
2. Rastrea cada blob mediante el intento de encontrarlo en el frame actual y en el anterior.
3. Añade un nuevo blob a la lista de blobs rastreados si éste puede ser rastreado satisfactoriamente a través de múltiples frames sucesivos.

El segundo de ellos, **BD_Simple**, detecta un nuevo blob mediante movimiento uniforme de los componentes conectados de la máscara de FG.

A continuación se muestra la parte del código fuente de nuestra aplicación asociada al módulo "*Blob Entering Detection*"

```
/* list of BLOB DETECTION modules */
typedef struct DefModule_BlobDetector
{
    CvBlobDetector* (*create)();
    char* nickname;
    char* description;
} DefModule_BlobDetector;
DefModule_BlobDetector BlobDetector_Modules[] =
{
    {cvCreateBlobDetectorCC,"BD_CC","Detect new blob by tracking CC
of FG mask"},
    {cvCreateBlobDetectorSimple,"BD_Simple","Detect new blob by
uniform moving of connected components of FG mask"},
```

```
    {NULL,NULL,NULL}
};
```

Blob Tracking

En esta etapa, se asigna un ID a cada blob y se rastrea su movimiento frame-a-frame. El módulo de rastreo de blob proporciona rastreo frame-a-frame de la posición y tamaño del blob.

Nosotros disponemos de muchas alternativas para realizar esta labor, concretamente veinte módulos, pero se decide que, por su mejor comportamiento, hagamos uso en nuestra experimentación del módulo denominado **FUZZY**.

A continuación se muestra la parte del código fuente de nuestra aplicación asociada al módulo "*Blob Tracking*"

```
/* list of BLOB TRACKING modules */
typedef struct DefModule_BlobTracker
{
    CvBlobTracker* (*create)();
    char* nickname;
    char* description;
} DefModule_BlobTracker;
DefModule_BlobTracker BlobTracker_Modules[] =
{
    {cvCreateBlobTrackerCCMSPF,"CCMSPF","connected component
tracking and MSPF resolver for collision"},
    {cvCreateBlobTrackerCC,"CC","Simple connected component
tracking"},
    {cvCreateBlobTrackerUMDA,"UMDA","Association by UMDA"},
    {cvCreateBlobTrackerPBIL,"PBIL","Association by PBIL"},
    {cvCreateBlobTrackerCGA,"CGA","Association by CGA"},
    {cvCreateBlobTrackerFUZZY,"FUZZY","Association by FUZZY"},
    {cvCreateBlobTrackerGA,"GA","Association by GA"},
    {cvCreateBlobTrackerHC,"HC","Association by Hill Climbing"},
    {cvCreateBlobTrackerSA,"SA","Association by Simulated
Annealing"},
    {cvCreateBlobTrackerTABU,"TABU","Association by TABU Search"},
    {cvCreateBlobTrackerTABUINI,"TABUINI","Association by TABU
Search with Initialization"},
    {cvCreateBlobTrackerTABUSWAP,"TABUSWAP","Association by TABU
Search with SWAP"},
    {cvCreateBlobTrackerTABU_1,"TABU_1","Association by TABU
Search(individual evaluation)"},
    {cvCreateBlobTrackerTABU_RGB,"TABU_RGB","Association by TABU
Search (color evaluation)"},
    {cvCreateBlobTrackerTABU_HSV,"TABU_HSV","Association by TABU
Search (color evaluation using HSV planes)"},
    {cvCreateBlobTrackerMS,"MS","Mean shift algorithm "},
    {cvCreateBlobTrackerTABU_RGB_H,"TABU_RGB_H","Association by
TABU Search (color histogram evaluation)"},
    {cvCreateBlobTrackerMSFG,"MSFG","Mean shift algorithm with FG
mask using"},
```

```
    {cvCreateBlobTrackerMSPF,"MSPF","Particle filtering based on MS
weight"},
   {cvCreateBlobTrackerPF,"PF","Particle filtering"},
    {NULL,NULL,NULL}
};
```

Trajectory Generation

En esta etapa, se recogen las posiciones de los blobs y se guarda la trayectoria de cada uno de ellos en el disco duro.

En nuestra aplicación disponemos de dos alternativas para realizar esta labor, dos módulos, denominados **YML** y **RawTracks**. En nuestra experimentación haremos uso del primero, **YML**, que, como su propio nombre indica, genera un registro del rastreo en formato YML. El fichero en formato YML generado servirá posteriormente para mostrar los resultados del rastreo en formato XML, formato que permitirá más fácilmente llevar a cabo la evaluación de nuestros resultados.

A continuación se muestra la parte del código fuente de nuestra aplicación asociada al módulo "*Trajectory Generation*"

```
/* list of BLOB TRAJECTORY GENERATION modules */
typedef struct DefModule_BlobTrackGen
{
    CvBlobTrackGen* (*create)();
    char* nickname;
    char* description;
} DefModule_BlobTrackGen;
DefModule_BlobTrackGen BlobTrackGen_Modules[] =
{
    {cvCreateModuleBlobTrackGenYML,"YML","Generate track record in
YML format as synthetic video data"},
    {cvCreateModuleBlobTrackGen1,"RawTracks","Generate raw track
record (x,y,sx,sy),()... in each line"},
    {NULL,NULL,NULL}
};
```

Trajectory Post Processing

Esta etapa es opcional, como ya se dijo anteriormente. En ella lo que se hace es ejecutarse una función suavizada sobre una trayectoria de blob.

En nuestra aplicación vemos como se refleja esta situación, ya que se da la posibilidad de escoger el módulo **None**, para el caso en que no se desee realizar esta etapa. En caso contrario, tenemos otro módulo, denominado **Kalman**, que hace uso del método de los filtros de Kalman para la posición y tamaño del blob. En nuestro caso, a la hora de realizar la experimentación, haremos uso del primero.

A continuación se muestra la parte del código fuente de nuestra aplicación asociada al módulo "*Trajectory Post Processing*"

```
/* list of BLOB TRAJECTORY POST PROCESSING modules */
typedef struct DefModule_BlobTrackPostProc
{
    CvBlobTrackPostProc* (*create)();
    char* nickname;
    char* description;
} DefModule_BlobTrackPostProc;
DefModule_BlobTrackPostProc BlobTrackPostProc_Modules[] =
{
    {cvCreateModuleBlobTrackPostProcKalman,"Kalman","Kalman
filtering of blob position and size"},
    {NULL,"None","No post processing filter"},
//
{cvCreateModuleBlobTrackPostProcTimeAverRect,"TimeAverRect","Averag
e by time using rectangle window"},
//
{cvCreateModuleBlobTrackPostProcTimeAverExp,"TimeAverExp","Average
by time using exponential window"},
    {NULL,NULL,NULL}
};
```

Trajectory Analysis

En esta etapa, se realiza un análisis de las trayectorias de los blobs. De esta forma se pueden detectar trayectorias anormales.

Nosotros disponemos de muchas alternativas para realizar esta labor, concretamente siete módulos.

Por un lado, se da la posibilidad de no realizar esta tarea mediante el módulo denominado **None**.

Por otro lado, existen otros basados en histogramas. De esta manera, para detectar trayectorias anormales, se usan histogramas de enfoque de diversas dimensiones. Estos módulos tratan una trayectoria como un conjunto independiente de vectores característicos. Cada vector característico incluye características tales como posición del blob, velocidad del blob, y estado de duración del blob. El histograma de estas características es continuamente recogido y analizado. Así, si el blob actual tiene características que fueron nunca o raramente observadas anteriormente, entonces el blob y sus trayectorias son clasificadas como anormales.

También existen otros módulos, concretamente dos, que no están basados en histogramas.

De todas formas, debido a que este aspecto no es fundamental en nuestro trabajo, se optará en la experimentación por no realizar esta labor mediante la selección del módulo **None**.

A continuación se muestra la parte del código fuente de nuestra aplicación asociada al módulo "*Blob Tracking*"

```
/* list of BLOB TRAJECTORY ANALYSIS modules */
CvBlobTrackAnalysis* cvCreateModuleBlobTrackAnalysisDetector();

typedef struct DefModule_BlobTrackAnalysis
{
    CvBlobTrackAnalysis* (*create)();
    char* nickname;
    char* description;
} DefModule_BlobTrackAnalysis;
DefModule_BlobTrackAnalysis BlobTrackAnalysis_Modules[] =
{
    {cvCreateModuleBlobTrackAnalysisHistPVS,"HistPVS","Histogramm
of 5D feture vector analysis (x,y,vx,vy,state)"},
    {NULL,"None","No trajectory analiser"},
    {cvCreateModuleBlobTrackAnalysisHistP,"HistP","Histogramm of 2D
feture vector analysis (x,y)"},
    {cvCreateModuleBlobTrackAnalysisHistPV,"HistPV","Histogramm of
4D feture vector analysis (x,y,vx,vy)"},
    {cvCreateModuleBlobTrackAnalysisHistSS,"HistSS","Histogramm of
4D feture vector analysis (startpos,endpos)"},
    {cvCreateModuleBlobTrackAnalysisTrackDist,"TrackDist","Compare
tracks directly"},
    {cvCreateModuleBlobTrackAnalysisIOR,"IOR","Integrator (by OR
operation) of several analysers "},
    {NULL,NULL,NULL}
};
```

3.3. Experimentación con FGD y MOG

Hasta ahora, con la explicación dada en el apartado anterior, tenemos una idea de cómo funciona nuestra aplicación. Hemos visto por encima cada uno de los módulos que componen el sistema, sin llegar a profundizar en ninguno de ellos. Ahora bien, será en este apartado donde nos metamos de lleno con el primero de los módulos, el referente a los algoritmos de segmentación de background y foreground, que es verdaderamente el principal objetivo de este trabajo.

Una vez que ya tenemos una idea de cómo funciona nuestra aplicación, la siguiente cuestión de interés será detallar todo el proceso de experimentación. Para ello, lo primero que haremos será centrarnos en los parámetros que tienen asociados los dos algoritmos,

viendo cuáles son y el significado que tienen. A continuación, iremos experimentando con cada uno de los vídeos de los que disponemos para concluir qué algoritmo es el más adecuado y con qué parámetros, y haciendo una comparación de cómo se comportan los dos algoritmos en los puntos clave del vídeo.

Parámetros asociados a FGD y MOG

Comenzaremos con el primero de los dos algoritmos, el algoritmo FGD. Durante la explicación teórica que se hizo sobre este algoritmo se pudieron ver parámetros que utilizaba, así como valores que eran adecuados. Pero ahora, la idea es mostrar todos los parámetros que tiene en nuestra aplicación, y que serán los que haya que ir estudiando a lo largo de esta etapa de experimentación. Para ello, se muestra a continuación la parte del código fuente donde se ve cuáles son estos parámetros:

```
typedef struct CvFGDStatModelParams
{
    int           Lc, N1c, N2c, Lcc, N1cc, N2cc,
is_obj_without_holes, perform_morphing;
    float         alpha1, alpha2, alpha3, delta, T, minArea;
}
CvFGDStatModelParams;
```

A continuación, mostramos una tabla donde se refleja el significado que tiene cada uno de ellos:

Parámetro	Significado
Lc	Niveles cuantizados para el componente de color. Potencia de dos, normalmente 32, 64 o 128.
N1c	Número de vectores de color usados para modelar la variación de color del background normal en un píxel dado.
N2c	Número de vectores de color mantenidos en un píxel dado. Tiene que ser mayor que N1c, normalmente alrededor de 5/3 de N1c. Usado para permitir a los primeros N1c vectores adaptarse a cambios de background que ocurran con el paso del tiempo.
Lcc	Niveles cuantizados para el componente de co-ocurrencia de color. Potencia de dos, normalmente 16, 32 o 64.
N1cc	Número de vectores de co-ocurrencia de color usados para modelar la variación de color del background normal en un píxel dado.

N2cc	Número de vectores de co-ocurrencia de color mantenidos en un píxel dado. Tiene que ser mayor que N1cc, normalmente alrededor de 5/3 de N1cc. Usado para permitir a los primeros N1cc vectores adaptarse a cambios de background que ocurran con el paso del tiempo.
is_obj_without_holes	Si tiene valor TRUE se ignoran agujeros dentro de los blobs de foreground. Por defecto valor igual a TRUE.
perform_morphing	Número de iteraciones de limpieza sobre los blobs de foreground que se deforman continuamente. El valor por defecto para este parámetro es 1.
alpha1	Indica cómo de rápido se olvidan los valores de un píxel viejo de foreground. Normalmente este parámetro es puesto a 0.1.
alpha2	"Controles de velocidad de la función de aprendizaje". Depende del parámetro T. Su valor normalmente ronda el 0.005.
alpha3	Suplente de alpha2, usado (por ejemplo) para una rápida convergencia inicial. Su valor típico es el de 0.1.
delta	Afecta a la cuantización de color y co-ocurrencia de color. Este parámetro normalmente es puesto a 2.
T	"Un valor porcentual que determina cuando nuevas características pueden ser reconocidas como nuevo background". Normalmente 0.9.
minArea	Descarta blobs de foreground cuya caja (área donde queda recogido un objeto de foreground) es más pequeña que este umbral.

Estos parámetros están inicializados en nuestra aplicación como se muestra en la siguiente porción del código fuente:

```
/* default paremeters of foreground detection algorithm */
#define  CV_BGFG_FGD_LC              64
#define  CV_BGFG_FGD_N1C             30
#define  CV_BGFG_FGD_N2C             50

#define  CV_BGFG_FGD_LCC             32
#define  CV_BGFG_FGD_N1CC            50
#define  CV_BGFG_FGD_N2CC            80

/* BG reference image update parameter */
#define  CV_BGFG_FGD_ALPHA_1         0.1f

/* stat model update parameter
```

```
   0.002f ~ 1K frame(~45sec), 0.005 ~ 18sec (if 25fps and
absolutely static BG) */
#define  CV_BGFG_FGD_ALPHA_2         0.005f

/* start value for alpha parameter (to fast initiate statistic
model) */
#define  CV_BGFG_FGD_ALPHA_3         0.1f

#define  CV_BGFG_FGD_DELTA           2

#define  CV_BGFG_FGD_T               0.9f

#define  CV_BGFG_FGD_MINAREA         15.f

#define  CV_BGFG_FGD_BG_UPDATE_TRESH 0.5f
```

De forma análoga, vemos los parámetros asociados al segundo de los algoritmos, el algoritmo MOG. En primer lugar, mostramos la parte del código fuente donde se define la estructura con los parámetros de dicho algoritmo:

```
typedef struct CvGaussBGStatModelParams
{
    int     win_size;               /* = 1/alpha */
    int     n_gauss;
    double  bg_threshold, std_threshold, minArea;
    double  weight_init, variance_init;
}CvGaussBGStatModelParams;
```

A continuación, mostramos una tabla donde reflejamos el significado que tiene cada uno de ellos:

Parámetro	Significado
win_size	Determina el valor de la constante de aprendizaje (alpha), ya que alpha=1/win_size.
n_gauss	Número de distribuciones gaussianas en la mezcla. Normalmente, su valor oscila entre 3 y 5.
bg_threshold	Umbral suma de los pesos para los test de background.
std_threshold	Umbral de desviación estándar (lambda). Normalmente su valor es 2.5
minArea	Descarta blobs de foreground cuya caja (área donde queda recogido un objeto de foreground) es más pequeña que este umbral.
weight_init	Peso inicial de la distribución gaussiana.
variante_init	Desviación estándar inicial de la distribución gaussiana.

Estos parámetros están inicializados en nuestra aplicación como se muestra en la siguiente porción del código fuente:

```
/* default parameters of gaussian background detection algorithm */
#define CV_BGFG_MOG_BACKGROUND_THRESHOLD     0.7     /* threshold
sum of weights for background test */
#define CV_BGFG_MOG_STD_THRESHOLD            2.5     /* lambda=2.5
is 99% */
#define CV_BGFG_MOG_WINDOW_SIZE              200     /* Learning
rate; alpha = 1/CV_GBG_WINDOW_SIZE */
#define CV_BGFG_MOG_NGAUSSIANS               5       /* = K =
number of Gaussians in mixture */
#define CV_BGFG_MOG_WEIGHT_INIT              0.05
#define CV_BGFG_MOG_SIGMA_INIT               30
#define CV_BGFG_MOG_MINAREA                  15.f


#define CV_BGFG_MOG_NCOLORS                  3
```

Experimentación

A continuación se detallará todo el proceso de experimentación que se ha realizado sobre la batería de vídeos de la que se dispone. La idea es separar cada uno de los diferentes entornos en los que se ha trabajado, e ir viendo, para cada unos de ellos, los diferentes vídeos sobre los que se experimentó. De cada vídeo realizaremos una breve descripción, señalando los puntos clave, e indicaremos cuáles son los mejores parámetros para cada uno de los dos algoritmos, así como las diferencias más relevantes observadas entre ellos.

Pasamos a enumerar los diferentes entornos sobre los que se ha realizado la experimentación:

1) Interior de un edificio.
2) Campus universitario.
3) Centro comercial.
4) Estación de tren.
5) Aeropuerto

A continuación detallamos cada uno de los entornos, incluyendo en cada uno de ellos una serie de vídeos que nos sirva para analizar el comportamiento de los dos algoritmos.

➢ Interior de un edificio: Se trata del primer entorno sobre el que se ha trabajado. Si bien no queda claro del todo el entorno concreto del que se trata, al contrario que ocurre con el resto, sí podemos apreciar que se trata de un entorno de interior, ya que se observa mobiliario típico de una vivienda, al igual que se aprecia en el suelo zonas de mayor iluminación provocadas, seguramente, por el paso de la luz del sol por algún tipo de ventana. A continuación, mostramos una imagen del entorno en cuestión:

A continuación, nos centraremos en los vídeos asociados a este entorno:

Nombre del vídeo: *Browse1*

Descripción del vídeo: Durante el transcurso del vídeo se pueden observar a diferentes personas. Algunas de ellas están prácticamente estáticas, realizando escasos movimientos en zonas muy reducidas. Otras aparecen en la escena caminando, bien de forma continuada (sin pararse), o bien realizando paradas de corta duración.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30	*NG*= 3

N2c= 50 *N1cc*= 50 *N2cc*= 80	

Como se puede observar en la tabla, únicamente mostramos los parámetros específicos de los algoritmos. Para los módulos *Blob Entering Detection* y *Blob Tracking* utilizaremos, si no se dice lo contrario, los módulos **BD_CC** y **FUZZY**, respectivamente. Esto servirá para todos los vídeos sobre los que se experimente.

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo.

Antes nada, hay que señalar que el resultado de MOG presenta más ruido que el resultado de FGD, es decir, puntos que son detectados como foreground sin serlo.

- *Foreground de movimiento reducido*. Incluimos en este punto a las personas que se mueven en una región bastante pequeña, por lo que sus movimientos, en ocasiones son poco apreciables. En este caso, se prefiere el comportamiento de **MOG**, ya que es capaz de detectar a estas personas de mejor manera (de forma más continuada) que el otro algoritmo.
- *Foreground de movimiento continuado*. Incluimos en este punto a las personas que caminan sin detenerse en ningún instante. En este caso, hay que realizar un par de consideraciones. En primer lugar, con la resolución por defecto que tenemos del vídeo, se concluye que la detección de estas personas es mejor con el algoritmo MOG, ya que las detecta independientemente de la distancia a la que se encuentren de la cámara, mientras que el algoritmo FGD presenta dificultades para detectar a las personas que caminan alejadas de la cámara. Este aspecto se puede solucionar aumentando la resolución del vídeo (mediante una herramienta que varía la resolución de las imágenes que componen el vídeo). Y en segundo lugar, decir que la detección con el algoritmo FGD es más "perfecta", ya que delimita mejor la figura de la persona y no detecta la sombra, cosa que no ocurre con el algoritmo MOG. Por tanto, teniendo en cuenta los dos casos, y sabiendo que uno tiene solución, para este tipo de foreground se prefiere el algoritmo **FGD** (para este vídeo en concreto, donde el aumento de la resolución del vídeo soluciona el problema. En caso de trabajarse con la resolución por defecto se preferirá el algoritmo **MOG**).
- *Foreground en movimiento con paradas*. Incluimos en este punto a las personas que caminan y realizan una parada, no muy prolongada, para posteriormente continuar caminando. Es mejor el comportamiento que presenta el algoritmo **MOG**, ya que cuando estas personas se detienen, las mantiene como foreground, mientras que con

el otro algoritmo rápidamente las considera background (considera background la región del cuerpo cuyo color no contrasta en exceso con el color del background).

Nombre del vídeo: *Browse2*

Descripción del vídeo: Durante el transcurso del vídeo se pueden observar claramente a dos personas. Una de ellas sale caminando de la zona más cercana de la cámara, realiza una parada, y sale de la escena por el mismo lugar que apareció. La otra sale a escena de la zona más alejada, para ir caminando en dirección a la cámara, realiza una parada de mayor duración, retoma su movimiento, y sale de la escena.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de FGD presenta problemas de detección de foreground erróneo, como es la detección como foreground de la planta y de regiones en el suelo.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente. Hay que destacar el mejor comportamiento del algoritmo **MOG** respecto a las personas situadas lejos de la cámara.

- *Foreground en movimiento con paradas*. Para este vídeo en concreto, vemos como con **FGD** tiene un comportamiento diferente a lo dicho para el vídeo anterior, ya que al realizar la parada una persona, ésta sigue siendo considerada foreground. Es más que probable que el color de la persona (en este caso contrasta mucho con el color del background) tenga mucho que ver. Con el algoritmo **MOG**, se observa el mismo comportamiento descrito anteriormente.

Nombre del vídeo: *Browse3*

Descripción del vídeo: Durante el transcurso del vídeo se pueden observar claramente a varias personas. Una de ellas aparece al comienzo del vídeo parada en la zona más cercana de la cámara, y sale caminando de la escena. Otra sale a escena de la zona más alejada, para ir caminando en dirección a la cámara, realiza una parada, retoma su movimiento, y sale de la escena. También se observar la presencia de personas con movimientos reducidos, como vimos en algún vídeo anterior, así como dos personas que aparecen caminando en el piso superior (zona alejada de la cámara).

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de FGD presenta problemas de detección de foreground erróneo, como es la detección como foreground de la planta y de regiones en el suelo.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente. Hay que destacar el mejor comportamiento del algoritmo **MOG** respecto a las personas situadas lejos de la cámara.
- *Foreground en movimiento con paradas*. Con ambos algoritmos tiene un comportamiento similar, ya que durante el tiempo que la persona está parada (parada no muy larga) sigue siendo considerada foreground. Si bien es cierto que para el algoritmo FGD influye el color de la persona para que ésta siga siendo detectada en su totalidad, o sólo alguna parte de ella (la que más contrasta con el background).
- *Foreground de movimiento reducido*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento inicialmente parado*. Incluimos en este punto a los objetos de foreground que aparecen en la escena parados una vez que comienza el vídeo, y pasado un tiempo comienzan a moverse. Con FGD se detecta el objeto como foreground cuando éste comienza su movimiento. Con MOG esta detección se produce algo antes y, además, la región inicialmente ocupada por el objeto es detectada como foreground durante poco tiempo después de desaparecer de ahí el objeto que la ocupaba. Este comportamiento se puede considerar como normal, ya que cuando se inicia el vídeo y se encuentra una persona parada, ésta puede ser considerada como background, en el momento que ésta se desplaza se la detecta como foreground, y la región donde se encontraba es considerada también foreground (al presentar una apariencia diferente a la inicial) hasta que la nueva región es asimilada como background. Este comportamiento también se produce con el algoritmo FGD cuando aumentamos la resolución del vídeo (para lograr una mejor detección de los objetos lejanos a la cámara), tardando más tiempo en asimilar como background la región inicialmente ocupada por el objeto parado.

Nombre del vídeo: *Browse_WhileWaiting1*

Descripción del vídeo: Durante el transcurso del vídeo se pueden observar claramente a varias personas caminando. Algunas de ellas lo hacen en zonas más alejadas de la cámara, y otras en zonas más cercanas. No presenta ninguna novedad frente a los vídeos tratados anteriormente.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de FGD presenta problemas de detección de foreground erróneo, como es la detección como foreground de la planta y de regiones en el suelo.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente. Hay que destacar el mejor comportamiento del algoritmo **MOG** respecto a las personas situadas lejos de la cámara.

Nombre del vídeo: *browse_WhileWaiting2*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia principalmente a una persona. Se puede decir que ésta se comporta de una forma diferente a lo visto en vídeos anteriores, ya que combina diferentes casos estudiados. Inicialmente está parada, y no comienza a caminar pasado un buen rato, camina durante poco tiempo realizando paradas breves, simulando el comportamiento de una persona que se encuentra esperando a otra.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG

Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de FGD presenta problemas de detección de foreground erróneo, como es la detección como foreground de la planta y de regiones en el suelo.

- *Foreground en movimiento inicialmente parado*. Para este punto nos sirve lo dicho anteriormente. Hay que tener en cuenta que el tiempo que se considera foreground la región inicialmente ocupada por el objeto parado depende del tiempo que ésta esté parada, así, cuanto más tiempo permanezca el objeto parado más tiempo tardará en asimilar como background dicha región.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente. Es preferible en este entorno el uso de **MOG**, ya que el otro algoritmo se ve afectado tanto por los contrastes en los colores entre el foreground y el background, como por la distancia de la persona a la cámara (dependerá de la resolución del vídeo).

➢ Campus universitario: Se trata de un entorno de exterior, concretamente del exterior de un campus universitario, donde se puede apreciar una zona asfaltada para el tránsito y aparcamiento de vehículos, una zona ajardinada perteneciente al campus, los propios edificios pertenecientes a la universidad, entre otros objetos que iremos viendo en los diferentes vídeos relativos a este entorno. Hay que señalar que hay varias cámaras, colocadas en lugares diferentes, asociadas a este entorno. A continuación, mostramos las imágenes asociadas a los cámaras del entorno en cuestión:

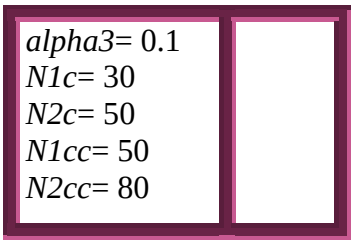
A continuación, nos centraremos en los vídeos asociados a este entorno:

Nombre del vídeo: *camera1* (PETS2001-DATASET1-TESTING)→Primera imagen.

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas y vehículos, con un comportamiento similar al visto en el anterior entorno, es decir, moviéndose de forma continuada o realizando paradas. Como novedades se puede señalar la presencia de background dinámico, como es el movimiento de las ramas de un árbol o el cambio de color en la superficie del edificio provocada por el cambio de iluminación. También se aprecia como algunos de los objetos en movimiento se cruzan, así como grupos de personas moviéndose de manera conjunta.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005	*NG*= 3



Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de MOG presenta ruido.

- *Foreground de movimiento continuado*. Se prefiere el comportamiento del algoritmo **MOG**, ya que la detección de este tipo de foreground no se ve afectada por la distancia y color de los objetos (aunque la detección no sea tan "perfecta" y también se detecten las sombras de los objetos).
- *Foreground en movimiento con paradas*. Con ambos algoritmos se observa un comportamiento similar, es decir, se detecta el objeto hasta que se detiene, y una vez parado va poco a poco asimilando esa región como background. Aunque no se aprecie gran diferencia, sí que da la impresión que con el algoritmo MOG tarda algo más en asimilar la región como background.
- *Foreground en movimiento cruzándose*. Incluimos en este punto a la situación en la que personas (en general, objetos) que caminan en sentido contrario se cruzan. Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que se cruzan como si ese cruce no se produjera. Por tanto, a la hora de decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado.
- *Foreground en movimiento agrupado.* Incluimos en este punto a personas que van caminando en grupo, es decir, objetos que se mueven conjuntamente. Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que componen el grupo, son tratados como objetos independientes. Por tanto, a la hora de decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado.
- *Background dinámico*. Incluimos en este punto a la parte del background no estacionaria. En este vídeo en particular se puede apreciar el movimiento de las ramas de un árbol o el cambio de color en la superficie del edificio provocada por el cambio de

iluminación. Pues bien, el comportamiento mejor se produce con el algoritmo **FGD**, ya que lo considera correctamente como background. Sin embargo, esos cambios en el background son considerados por el otro algoritmo, el MOG, como objetos de foreground, y no los vuelve a considerar background hasta que no los asimila como tal.

Nombre del vídeo: *camera2* (PETS2001-DATASET1-TESTING)→Segunda imagen

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que en el vídeo anterior, ya que ambos graban lo mismo, pero desde diferentes cámaras.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de MOG presenta ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.

- *Background dinámico*. Para este punto nos sirve lo dicho anteriormente.

Nombre del vídeo: *camera1* (PETS2001-DATASET1-TRAINING)→Primera imagen.

Descripción del vídeo: Durante el transcurso del vídeo se pueden observar diferentes elementos de interés. Por un lado vuelve a presentarse background dinámico, reflejado en el movimiento de las ramas de un árbol y en cambios de iluminación en la superficie del edificio. También aparecen en la escena vehículos, realizando aparcamientos o poniéndose en marcha (tras un tiempo estacionado), así como personas caminando.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de MOG presenta ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. En este caso se trata de una parada muy breve que hace un vehículo como parte de la maniobra de aparcamiento. En este caso, no se llega a perder la detección del objeto. Se concluye que, para que un objeto de foreground que se detenga, es necesario que se mantenga un tiempo determinado

para que sea asimilado como nueva parte del background. Esto es aplicable a los dos algoritmos.

- *Foreground en movimiento inicialmente parado*. Con FGD se detecta el objeto como foreground cuando éste comienza su movimiento. Con MOG esta detección se produce algo antes y, además, la región inicialmente ocupada por el objeto es detectada como foreground durante poco tiempo después de desaparecer de ahí el objeto que la ocupaba. Se corresponde, por lo tanto, con el comportamiento observado en el entorno anterior.
- *Background dinámico*. Para este punto nos sirve lo dicho anteriormente.

Nombre del vídeo: *camera2* (PETS2001-DATASET1-TRAINING)→Segunda imagen.

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que en el vídeo anterior, ya que ambos graban lo mismo, pero desde diferentes cámaras.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de MOG presenta ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.

- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento inicialmente parado*. Para este punto nos sirve lo dicho anteriormente.
- *Background dinámico*. Para este punto nos sirve lo dicho anteriormente.

Nombre del vídeo: *camera1* (PETS2001-DATASET2-TESTING)→Tercera imagen.

Descripción del vídeo: Durante el transcurso del vídeo se aprecian diferentes elementos de interés. Por un lado vuelve a presentarse background dinámico (además del movimiento de las ramas de un árbol, se aprecia también el movimiento de nubes). Por otro lado, aparecen en la escena diferentes objetos de foreground: personas caminando, personas circulando en bicicleta y vehículos (algunos de ellos de forma continuada, otro realizando paradas propias de una maniobra de aparcamiento). Por último, decir que se produce algún cruce (solapamiento) de estos objetos de foreground.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de MOG presenta ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente. Recordamos la detección más "perfecta" del algoritmo FGD, y los problemas que tiene este mismo algoritmo en detectar objetos que según se van alejando se hacen de tal tamaño que no son detectados.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente. Recordamos que con paradas breves, no se pierde la detección del foreground. Comportamiento similar con los dos algoritmos.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente. Recordamos que este punto no presenta ninguna complicación para los dos algoritmos.
- *Background dinámico*. Para este punto nos sirve lo dicho anteriormente. Recordamos el comportamiento satisfactorio de FGD.

➢ Centro comercial: Se trata de un entorno de interior, concretamente del interior de un centro comercial. Se ve cómo la cámara está situada en el escaparate de una tienda, y está dirigida a grabar lo que ocurra en el exterior de la misma, es decir en los pasillos cercanos a la tienda. A lo largo de los vídeos que iremos detallando, se verán los elementos que formarán parte de la escena, pero es importante tener en cuenta que, como novedad, se podrán apreciar movimientos que ocurran en el interior de la tienda, ya que éstos se reflejarán en el cristal del escaparate. A continuación, mostramos la imagen asociada al entorno en cuestión:

A continuación, nos centraremos en los vídeos asociados a este entorno:

Nombre del vídeo: *1* (PETS2002-TESTING-DATASET1)

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas en los pasillos, con un comportamiento similar al visto en los entornos anteriores, es decir, moviéndose de forma continuada o realizando paradas (por ejemplo, delante del escaparate). Como novedad se puede señalar la presencia de movimientos reflejados en el cristal. También se aprecia como algunos de los objetos en movimiento se cruzan, así como grupos de personas moviéndose de manera conjunta. No se observa la presencia de background dinámico.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de ambos algoritmos presenta ruido. En el caso de MOG el ruido es similar al visto en los otros entornos, mientras que con FGD da la impresión que el ruido está relacionado con la presencia del cristal del escaparate.

- *Foreground de movimiento continuado*. Se prefiere el comportamiento del algoritmo **MOG**, ya que la detección de este tipo de foreground no se ve afectada por la distancia de los objetos, a pesar de que la detección no sea tan “perfecta” y se detecte las sombras de los objetos.
- *Foreground en movimiento con paradas*. Con ambos algoritmos se observa un comportamiento similar, es decir, al tratarse de paradas no muy largas y que además las personas no se quedan totalmente quietas, se las detecta correctamente como foreground el tiempo que permanecen observando el escaparate.

- *Foreground en movimiento cruzándose*. Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que se cruzan como si ese cruce no se produjera. Por tanto, a la hora de decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado, o lo que es lo mismo, a qué distancia de la cámara se produce ese cruce.
- *Foreground en movimiento agrupado.* Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que componen el grupo, son tratados como objetos independientes. Por tanto, a la hora de decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado, o lo que es igual, a qué distancia de la cámara se mueve el grupo.

Nombre del vídeo: *2* (PETS2002-TESTING-DATASET2)

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas en los pasillos, con un comportamiento similar al visto en vídeo anterior. Como novedades se pueden señalar la presencia de una persona en la escena prácticamente parada cuando da comienzo el vídeo, y paradas más prolongadas en frente del escaparate.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. En este caso las paradas son algo más largas. De todas formas, sirve lo dicho en el vídeo anterior, ya que, aunque sean paradas más largas, éstas no son paradas como las que pueda realizar un vehículo, si no que las personas siguen realizando ligeros movimientos y, por lo tanto, afectan a que sigan siendo detectadas. Únicamente para zonas más estáticas, como puede ser el tronco de la persona, se llega a asimilar como background si se llega a un determinado tiempo sin moverse, pero pronto, cuando retoma el movimiento, se vuelve a considerar foreground.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento inicialmente parado*. En este punto nos encontramos con una variación respecto a los anteriores entornos. En este vídeo, cuando la persona que se encuentra parada inicialmente retoma su movimiento, es detectada de forma similar por los dos algoritmos. La diferencia se encuentra en el tiempo que ese espacio, inicialmente ocupado por la persona, es considerado como foreground, ya que con FGD tarda más tiempo en asimilar esa región como background (comportamiento contrario al visto en los otros entornos).

Nombre del vídeo: *3* (PETS2002-TESTING-DATASET3)

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas en los pasillos, con un comportamiento similar al visto en vídeos anteriores. Como novedad se pueden señalar la presencia de un mayor número de personas en el transcurso del vídeo, algunas de ellas concentradas en frente del escaparate. Vuelve a aparecer una persona en la escena prácticamente parada cuando da comienzo el vídeo, ésta en vez de desaparecer rápidamente de la escena (como ocurría en el vídeo anterior), permanece durante todo el vídeo compaginando paradas con breves desplazamientos.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento inicialmente parado*. Para este punto nos sirve lo dicho en el vídeo anterior, es decir, un mejor comportamiento con el uso del algoritmo **MOG**.

Nombre del vídeo: *1* (PETS2002-TRAINING-DATASET1)

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas en los pasillos, con un comportamiento similar al visto en vídeos anteriores. A pesar de la corta duración del vídeo, se pueden ver personas caminando en solitario, en grupo, cruzándose, incluso una persona que forma parte de la escena cuando comienza el vídeo. También se vuelven a ver personas reflejadas en el cristal del escaparate.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de ambos algoritmos presenta ruido (como ocurría en el primer vídeo asociado a este entorno).

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente. Podemos incluir también el movimiento de personas que se refleja en el cristal. Ambos algoritmos detectan satisfactoriamente a las personas reflejadas.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.

Nombre del vídeo: *2* (PETS2002-TRAINING-DATASET2)

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas en los pasillos, con un comportamiento similar al visto en vídeos anteriores. Se pueden ver personas caminando en solitario, personas que se cruzan, personas que se detienen a mirar el escaparate, y personas reflejadas en el cristal.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado de ambos algoritmos presenta ruido (como ocurre en otros vídeos asociados a este entorno).

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.

➢ Estación de tren: Se trata de un entorno de interior, concretamente del interior de una estación de tren. Se dispone de cuatro cámaras situadas en lugares diferentes, por lo que, aunque las cuatro graben lo mismo, en cada una se verán diferentes elementos de interés. En general, se puede diferenciar en la escena una zona de bar/restaurante, un pasillo donde transitará un gran número de personas, otra zona con asientos para la espera de los trenes, y una última zona donde se verá el andén y los trenes. Por el gran número de objetos de foreground que aparecen en la escena, éste puede ser considerado como el entorno más complejo de los vistos hasta ahora. A continuación, mostramos las cuatro imágenes asociadas a las cuatro cámaras del entorno en cuestión:

A continuación, nos centraremos en cuatro vídeos (video1, video2, video3 y video4) asociados a cada una de las imágenes mostradas de este entorno (en el orden que se muestran):

Nombre del vídeo: *video1*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas por el pasillo, con un comportamiento similar al visto en los entornos anteriores, esto es, moviéndose de forma continuada, realizando paradas, moviéndose en grupo, cruzándose… Algunas de ellas se ven claramente por aparecer cerca de la cámara, mientras que otras se pueden apreciar en la lejanía. También se puede observar a gente sentada, bien en el bar-restaurante o en los asientos ubicados cerca del andén. Como principal novedad respecto a entornos anteriores se puede mencionar la gran cantidad de gente que aparece en este vídeo. No se observa la presencia de background dinámico.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

- *Foreground de movimiento continuado*. Se prefiere el comportamiento del algoritmo **MOG**, ya que la detección de este tipo de foreground no se ve afectada por la distancia

de los objetos, a pesar de que la detección no sea tan "perfecta" y se detecte las sombras de los objetos. En este entorno en particular, este aspecto toma gran relevancia, ya que muchas personas caminan alejadas de la cámara. Comparando el resultado de los dos algoritmos, claramente es más satisfactorio el conseguido por MOG, que también detecta con mayor facilidad los reflejos de las personas.

- *Foreground en movimiento con paradas*. Con ambos algoritmos se observa un comportamiento similar, es decir, al tratarse de paradas no muy largas y que además las personas no se quedan totalmente quietas, se las mantiene como foreground el tiempo que permanecen sin moverse. En este vídeo, debido a la mejor detección del foreground por parte del algoritmo **MOG** (explicado en el punto anterior), se prefiere el uso de este algoritmo.
- *Foreground en movimiento cruzándose*. Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que se cruzan como si ese cruce no se produjera. Por tanto, a la hora de decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado.
- *Foreground en movimiento agrupado.* Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que componen el grupo, son tratados como objetos independientes. Por tanto, a la hora de decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado.
- *Foreground de movimiento reducido*. Es el caso de las personas que se encuentran sentadas esperando la llegada del tren. Éstas realizan ligeros movimientos mientras permanecen en el asiento, y son detectados con mayor facilidad por el algoritmo **MOG**, si bien es más que probable que este comportamiento está condicionado por la distancia a la que se encuentren de la cámara.

Nombre del vídeo: *video2*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que en el vídeo anterior, ya que ambos graban lo mismo, pero desde diferentes cámaras. En este caso, la cámara está más alejada, por lo que las personas que caminen en el pasillo serán menos visibles que en el vídeo anterior. También se observa como hay superficies que cambian su color debido a la variación de iluminación, por lo que volvemos a encontrarnos en este vídeo con background dinámico.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.
- *Foreground de movimiento reducido*. En este vídeo son ejemplo de este tipo de foreground algunos empleados de la estación que permanecen de pie realizando ligeros movimientos. Vuelve a confirmarse que el algoritmo **MOG** se comporta mejor.
- *Background dinámico*. Al igual que ocurría en entornos anteriores, se puede observar como el algoritmo **FGD** trata esta parte de background como tal, mientras que el otro algoritmo, el MOG, considera estos cambios que ocurren en el background como foreground.

Nombre del vídeo: *video3*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que en los dos vídeos anteriores, ya que los tres graban lo mismo, pero desde diferentes cámaras. En este caso, la cámara está orientada a grabar lo que ocurre en una zona reducida del pasillo y la zona del andén pegada a ella. También hay que señalar que la cámara está cercana al pasillo, más que en los dos vídeos anteriores, por lo que las personas que aparezcan en la escena serán más visibles.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

- *Foreground de movimiento continuado*. Debido a lo cercana que se encuentra la cámara se aprecia un mejor comportamiento del algoritmo FGD respecto al observado en otros vídeos. Aún así, se puede ver como la detección es mejor en el pasillo, zona más cercana a la cámara, que en el andén, más alejado del pasillo.
- *Foreground en movimiento con paradas*. Con la persona que realiza la parada, se puede apreciar como el algoritmo MOG la mantiene siempre como foreground, mientras que el algoritmo FGD tiene dificultades en seguir manteniéndola en su totalidad como foreground.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.

- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.

Nombre del vídeo: *video4*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que en los vídeos asociados a este entorno, ya que todos graban lo mismo, pero desde diferentes cámaras. En este caso, el vídeo presenta gran similitud con el anterior. La cámara está orientada a grabar lo que ocurre en una zona del pasillo algo más amplia que en el anterior, y también incluye la zona del andén pegada a ella y la zona de asientos para la espera de trenes.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

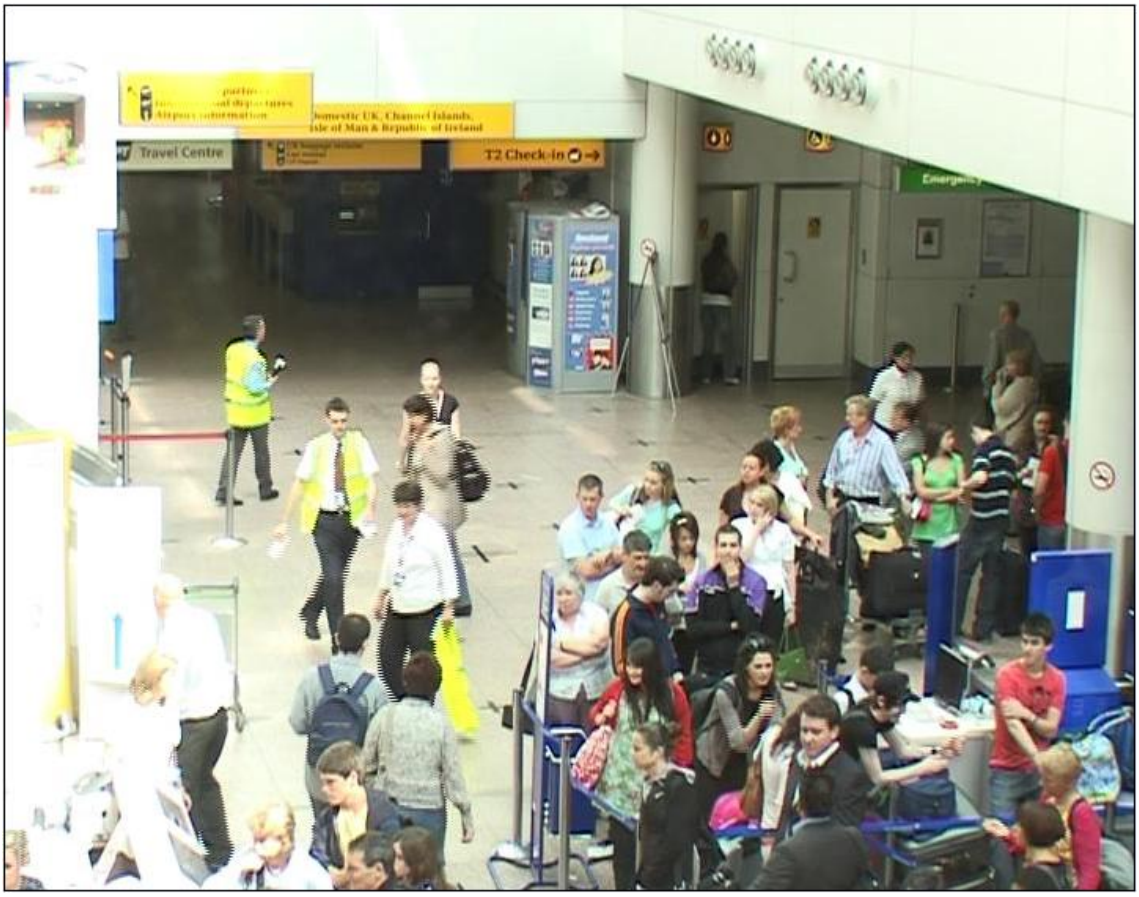
Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

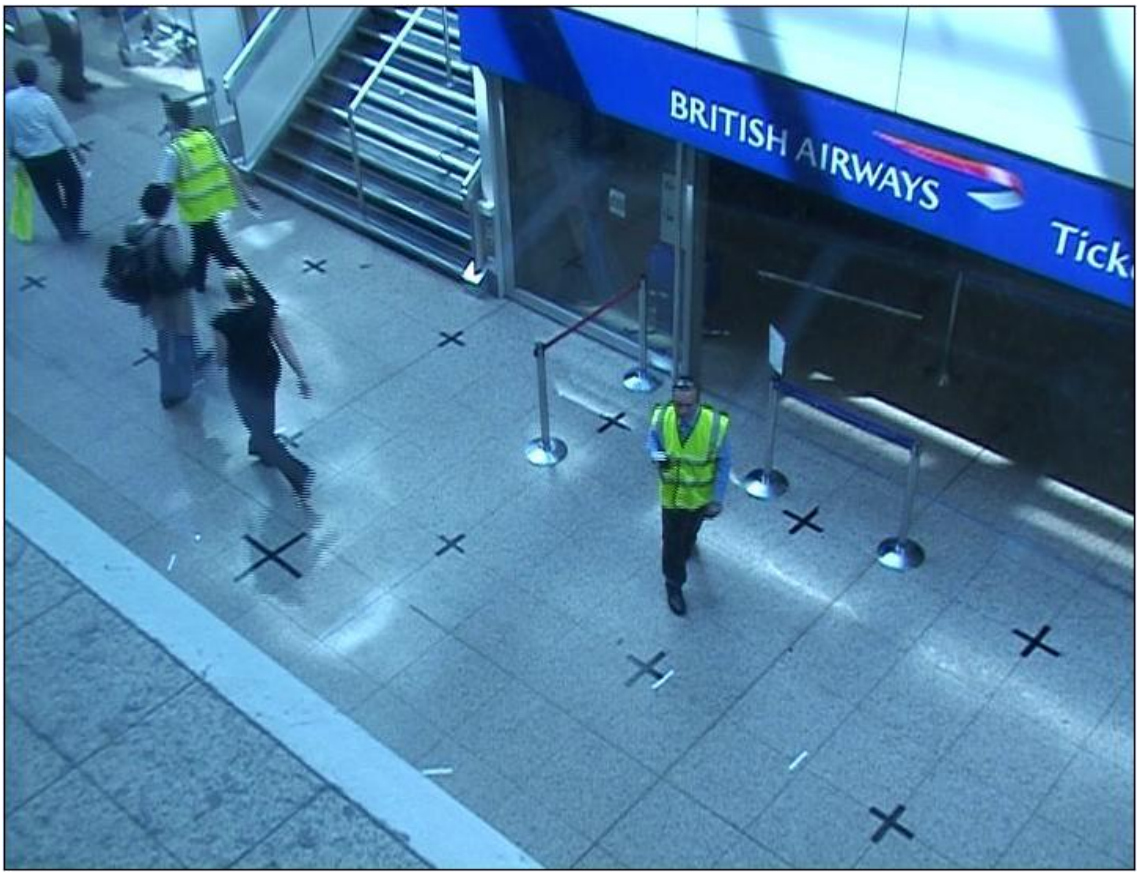
- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.

- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.
- *Foreground de movimiento reducido*. Para este punto nos sirve lo dicho anteriormente.

➢ Aeropuerto: Se trata de un entorno de interior, concretamente del interior de un aeropuerto. Se dispone de cuatro cámaras situadas en lugares diferentes, por lo que, aunque las cuatro graben lo mismo, en cada una se verán diferentes elementos de interés, que se irán viendo en los vídeos que se analicen a continuación. En general, se puede diferenciar en la escena un pasillo central donde transitan muchas personas, y una zona donde se encuentra gente esperando en una fila para poder embarcar. Por el gran número de objetos de foreground que aparecen en la escena, éste puede ser considerado, junto con el anterior, como el entorno más complejo de los vistos hasta ahora. A continuación, mostramos las cuatro imágenes asociadas a las cuatro cámaras del entorno en cuestión:




Travel Centre
T2 Check-in


BRITISH AIRWAYS

A continuación, nos centraremos en cuatro vídeos (video1, video2, video3 y video4) asociados a cada una de las imágenes mostradas de este entorno (en el orden que se muestran):

Nombre del vídeo: *video1*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia el tránsito de personas por el pasillo, con un comportamiento similar al visto en los entornos anteriores, esto es, moviéndose de forma continuada, realizando paradas, moviéndose en grupo, cruzándose… Algunas de ellas se ven claramente por aparecer cerca de la cámara, mientras que otras se pueden apreciar algo más lejanas, pero nunca tan lejanas como ocurría en el entorno anterior. También se puede observar a gente esperando en una fila para embarcar, siendo su movimiento muy reducido (los típicos movimientos de personas que se encuentran esperando en una fila). Al igual que ocurría con el entorno anterior se puede destacar la gran cantidad de gente que aparece en este vídeo. Por último, se observa la presencia de background dinámico, concretamente cambios de color en superficies del background provocadas por cambios de iluminación en la escena.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

- *Foreground de movimiento continuado*. Se prefiere el comportamiento del algoritmo **MOG**, ya que la detección de este tipo de foreground no se ve afectada por la distancia de los objetos, a pesar de que la detección no sea tan "perfecta" y se detecte las sombras de los objetos. También conviene señalar como con MOG la detección de los objetos es completa, es decir, que detecta al objeto como un único elemento, mientras que en ocasiones se observa como con el algoritmo FGD no se detecta al objeto en su totalidad, y sí por partes, lo que dificulta conocer si el objeto detectado es una persona u otra cosa.
- *Foreground en movimiento con paradas*. Con ambos algoritmos se observa un comportamiento similar, es decir, al tratarse de paradas no muy largas y que además las personas no se quedan totalmente quietas, se las mantiene como foreground el tiempo que permanecen sin moverse. En este vídeo, debido a la mejor detección del foreground por parte del algoritmo **MOG** (explicado en el punto anterior), se prefiere el uso de este algoritmo.
- *Foreground en movimiento cruzándose*. Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que se cruzan como si ese cruce no se produjera. Por tanto, a la hora de

decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado.

- *Foreground en movimiento agrupado.* Este aspecto no supone ningún problema para nuestros dos algoritmos, ya que ambos mantienen la detección de cada uno de los objetos que componen el grupo, son tratados como objetos independientes. Por tanto, a la hora de decidir cuál de los dos algoritmos se comporta mejor en este punto nos fijaremos en cómo se comportan con la detección de foreground de movimiento continuado.
- *Foreground de movimiento reducido*. Es el caso de las personas que se encuentran en la fila esperando a embarcar. Éstas realizan ligeros movimientos mientras permanecen en la fila, y son detectados con mayor facilidad por el algoritmo **MOG**.
- *Background dinámico*. En este vídeo concretamente, es mucha la superficie que cambia gradualmente su color debido a cambios de iluminación (en otros vídeos afectaba a una superficie menor), por lo que cuando el algoritmo MOG detecta esos cambios como foreground, el resultado de la experimentación se hace muy engorroso. Por otro lado, como viene siendo habitual, el algoritmo **FGD** no detecta el background dinámico como foreground.

Nombre del vídeo: *video2*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que en el vídeo anterior, ya que ambos graban lo mismo, pero desde diferentes cámaras. En este caso, la cámara está situada en la misma línea pero más cercana a la zona donde se encuentra la fila, como puede apreciarse en las imágenes relativas a las cámaras aportadas en la descripción del entorno. Con esta disposición de la cámara tan parecida a la anterior, se puede entender que no habrá grandes diferencias, se presentan, por lo tanto, las mismas situaciones que las vistas en el vídeo anterior.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30	*NG*= 3

N2c= 50 *N1cc*= 50 *N2cc*= 80	

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

- *Foreground de movimiento continuado*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.
- *Foreground de movimiento reducido*. Para este punto nos sirve lo dicho anteriormente.
- *Background dinámico*. Para este punto nos sirve lo dicho anteriormente.

Nombre del vídeo: *video3*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que en los dos vídeos anteriores, ya que los tres graban lo mismo, pero desde diferentes cámaras. En este caso, la cámara está orientada a grabar lo que ocurre en una zona reducida del pasillo. También hay que señalar que la cámara está cercana al pasillo, más que en los dos vídeos anteriores, por lo que las personas que aparezcan en la escena serán más visibles. Se vuelven a apreciar todas las situaciones vistas en los dos vídeos anteriores.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	**MOG**

Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

- *Foreground de movimiento continuado*. Debido a lo cercana que se encuentra la cámara se aprecia un mejor comportamiento del algoritmo FGD respecto al observado en otros vídeos.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.
- *Foreground de movimiento reducido*. En este vídeo no se alcanza a ver la fila donde se encuentra la gente esperando, pero sí tenemos ejemplo de este tipo de foreground. En concreto, se observa un grupo de personas que se encuentran paradas hablando, y su comportamiento es análogo al observado con la gente que se encontraba en la fila en otros vídeos. El comportamiento de los dos algoritmos con este tipo de foreground se corresponde con el comentado anteriormente.
- *Background dinámico*. En este vídeo, a la finalización del mismo, vemos como la mayor parte de la superficie se ve afectada por un cambio de color motivado por el cambio de iluminación en la escena. Cuando ocurre esto, vemos como el resultado de MOG llega a detectar casi toda la escena como foreground, siendo éste un claro ejemplo del problema que supone detectar erróneamente el background dinámico como foreground.

Nombre del vídeo: *video4*

Descripción del vídeo: Durante el transcurso del vídeo se aprecia la misma situación que la descrita en los vídeos asociados a este entorno, ya que todos graban lo mismo, pero desde diferentes cámaras. En este caso, la cámara está situada a la misma altura que las personas que transitan por el pasillo y en sentido opuesto a las dos primeras, como se puede observar en las imágenes aportadas a la hora de describir este entorno. Como novedad respecto a los anteriores vídeos, cabe señalar que también se llega a visualizar una planta superior donde también transita gente.

Resultado de la experimentación: En la siguiente tabla mostraremos cuáles son los mejores parámetros para cada uno de los dos algoritmos.

FGD	MOG
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80	*NG*= 3

Comparación entre FGD y MOG: Para realizar la comparación entre los dos algoritmos, tomaremos como referencia los puntos clave mencionados en la descripción del vídeo. Para los puntos que ya se hayan comentado en vídeos anteriores, con el objetivo de no repetir lo mismo, sólo nos extenderemos más cuando haya alguna variación respecto a lo dicho anteriormente.

Antes nada, hay que señalar que el resultado del algoritmo MOG presenta algo de ruido.

- *Foreground de movimiento continuado*. Respecto a la novedad que presentaba este vídeo, las personas que aparecen caminando en una segunda planta, hay que decir que el resultado con MOG es más satisfactorio. Se constata lo dicho en vídeos anteriores, con personas alejadas es mucho mejor el comportamiento que presenta **MOG**. Con FGD tiene problemas en detectar este tipo de situaciones.
- *Foreground en movimiento con paradas*. Para este punto nos sirve lo dicho anteriormente.

- *Foreground en movimiento cruzándose*. Para este punto nos sirve lo dicho anteriormente.
- *Foreground en movimiento agrupado.* Para este punto nos sirve lo dicho anteriormente.
- *Foreground de movimiento reducido*. Para este punto nos sirve lo dicho anteriormente.
- *Background dinámico*. Para este punto nos sirve lo dicho anteriormente.

3.4. Conclusiones de la experimentación

En este apartado se expondrán las diferencias más relevantes observadas entre los dos algoritmos una vez finalizada la experimentación en todos los entornos detallados en el apartado anterior. De esta forma, se pretenderá concluir para qué tipo de vídeos será mejor un algoritmo que otro, y cuáles serán los parámetros más adecuados. Para ello, se irán viendo diferencias en los aspectos más delicados, es decir, en aquellos que conllevan más dificultades de detección. Todos ellos estructurados según estén asociados al background estático, background dinámico o foreground.

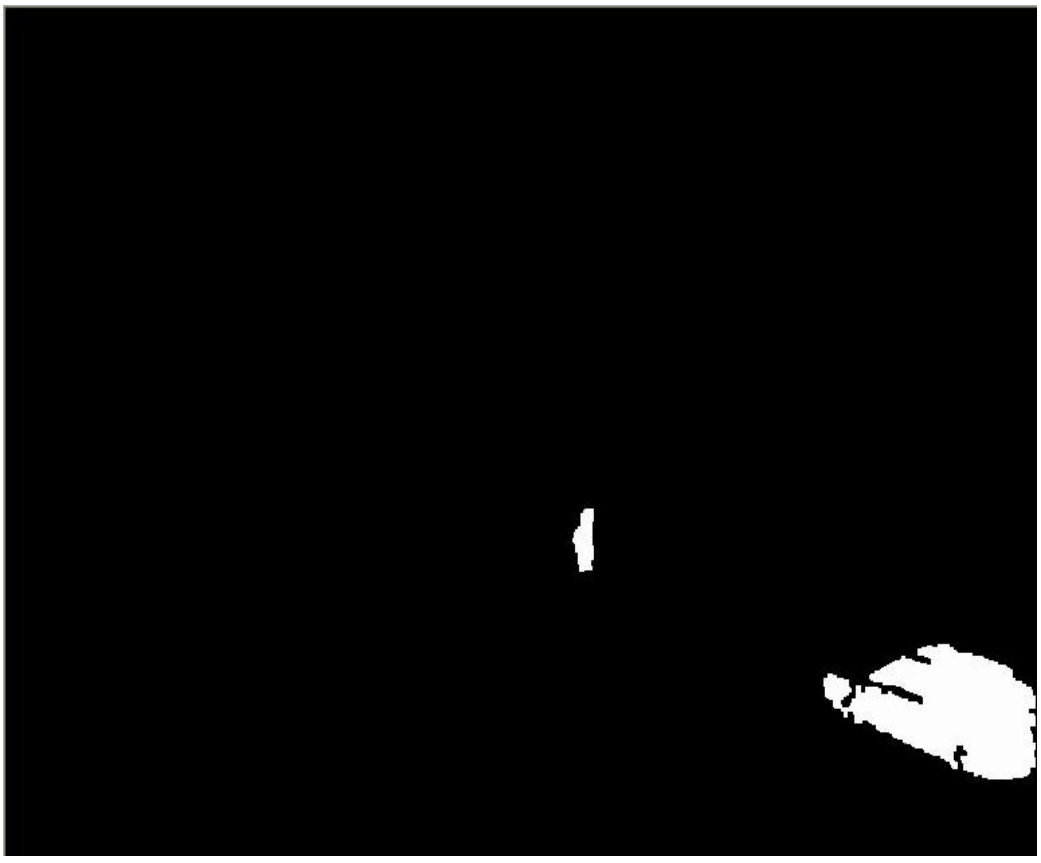
BACKGROUND:

- *Incorporación de background a la escena*: Consideramos en este punto a cualquier objeto que, pasado un tiempo, se incorpora a la escena como parte del background. Se puede entender como un tipo concreto de foreground en movimiento con paradas, donde el objeto en cuestión aparece en la escena moviéndose, llegando un momento en el que se detiene y no vuelve a reanudar su movimiento. Hay que tener en cuenta que podría darse el caso de que ese objeto volviera a moverse en algún momento, por lo que habría que estudiar también su comportamiento. Este caso, y por tanto el comportamiento, es análogo al que se produce cuando iniciado un vídeo se encuentra un objeto parado en la escena (considerándolo background) y posteriormente comienza a moverse. Este punto lo veremos más adelante.

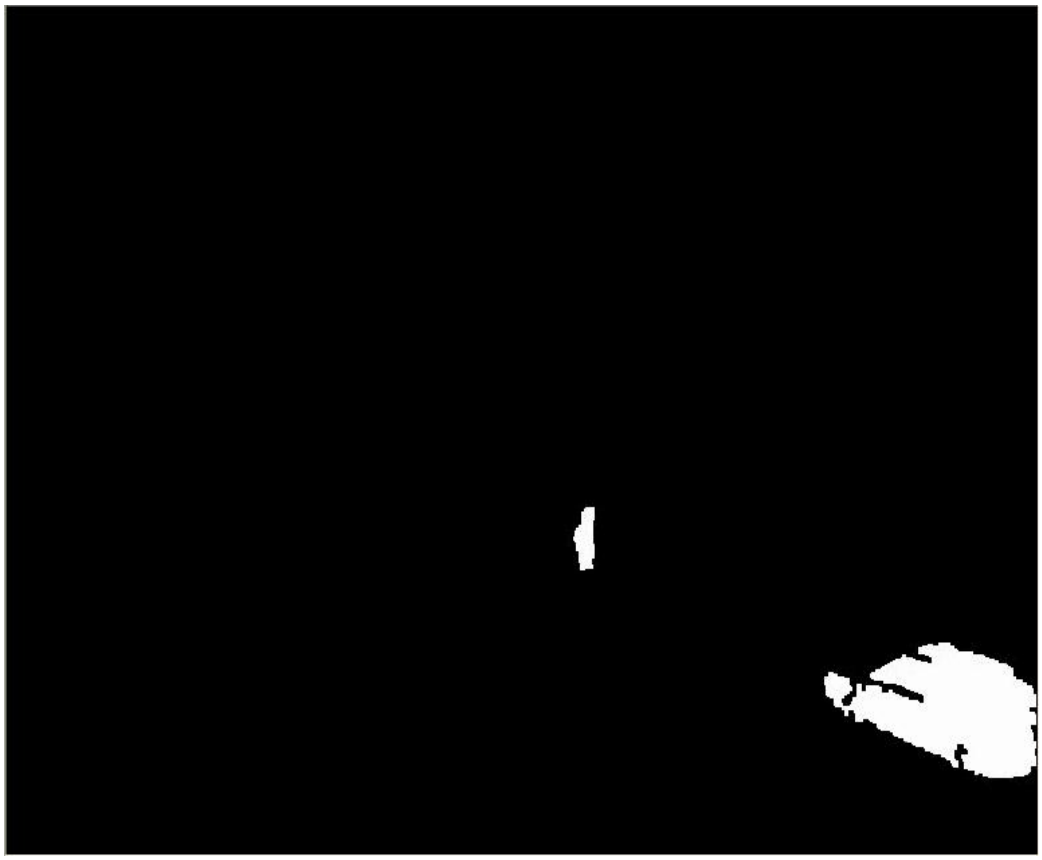
  Para ejemplificar este punto, vamos a fijarnos en un vehículo que aparece en la escena y realiza un aparcamiento. Una vez aparcado, el vehículo no volverá a moverse, por lo que pasará a formar parte del background. Reflejamos esta situación en las dos imágenes que mostramos a continuación. En la primera se observa cómo aparece el vehículo en la escena, y circula hasta quedar estacionado en la posición que muestra la segunda imagen.

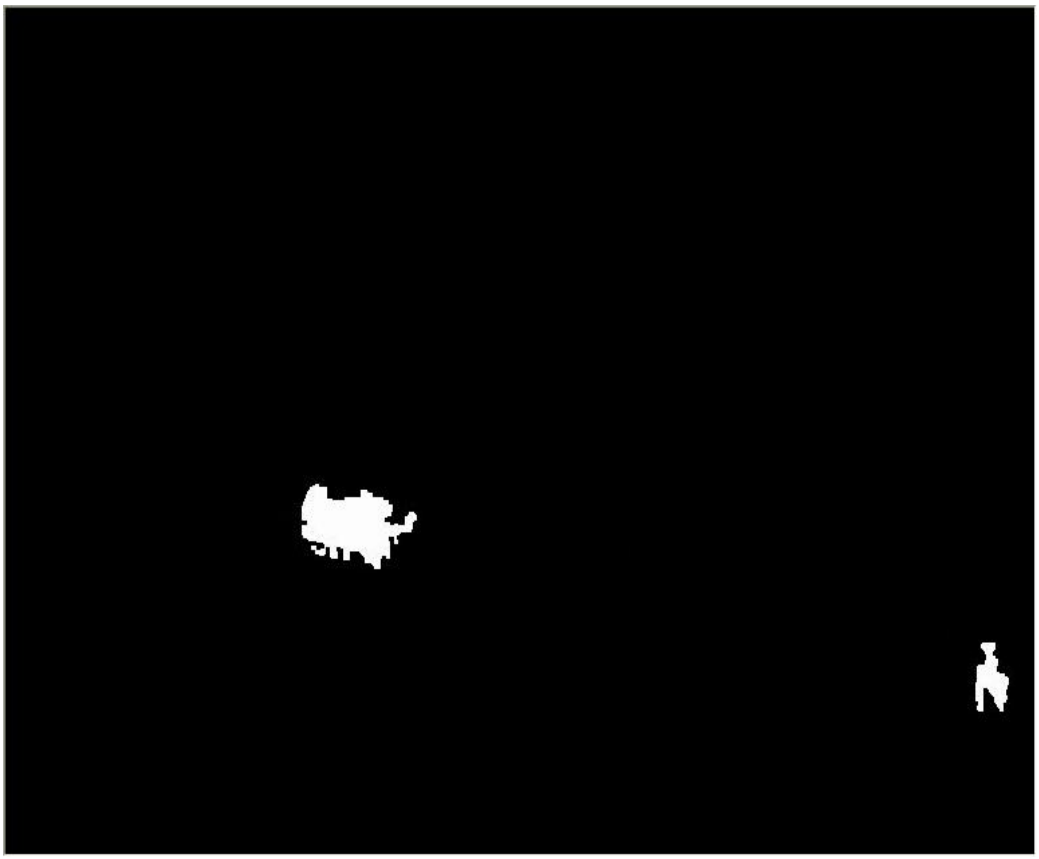
Con FGD el comportamiento es el deseado, concordando con lo que se vio en la explicación del algoritmo, ya que, cuando el objeto en cuestión deja de moverse en la escena, lo detecta durante un tiempo como foreground hasta que pase un período de tiempo determinado que lo haga asimilar como background nuevo. El parámetro que más influye en el tiempo que tarda en asimilar el objeto como background es *alpha2*. De esta forma, según se va aumentando su valor respecto del estándar, menos va tardando en asimilar como background la región ocupada por el vehículo. A continuación mostramos unas imágenes que reflejen esta situación. En la primera de ellas vemos como el vehículo es detectado cuando aparece en la escena (frame 0497). En la segunda veremos el momento en el que el vehículo se detiene (frame 0728). Y en la tercera, y última, veremos el instante en que el vehículo es completamente asimilado como background. Analizaremos el comportamiento para dos valores diferentes de *alpha2*: 0.005 (valor estándar) y 0.01 (valor mayor, y comentado también en la explicación del algoritmo).

alpha2 = 0.005



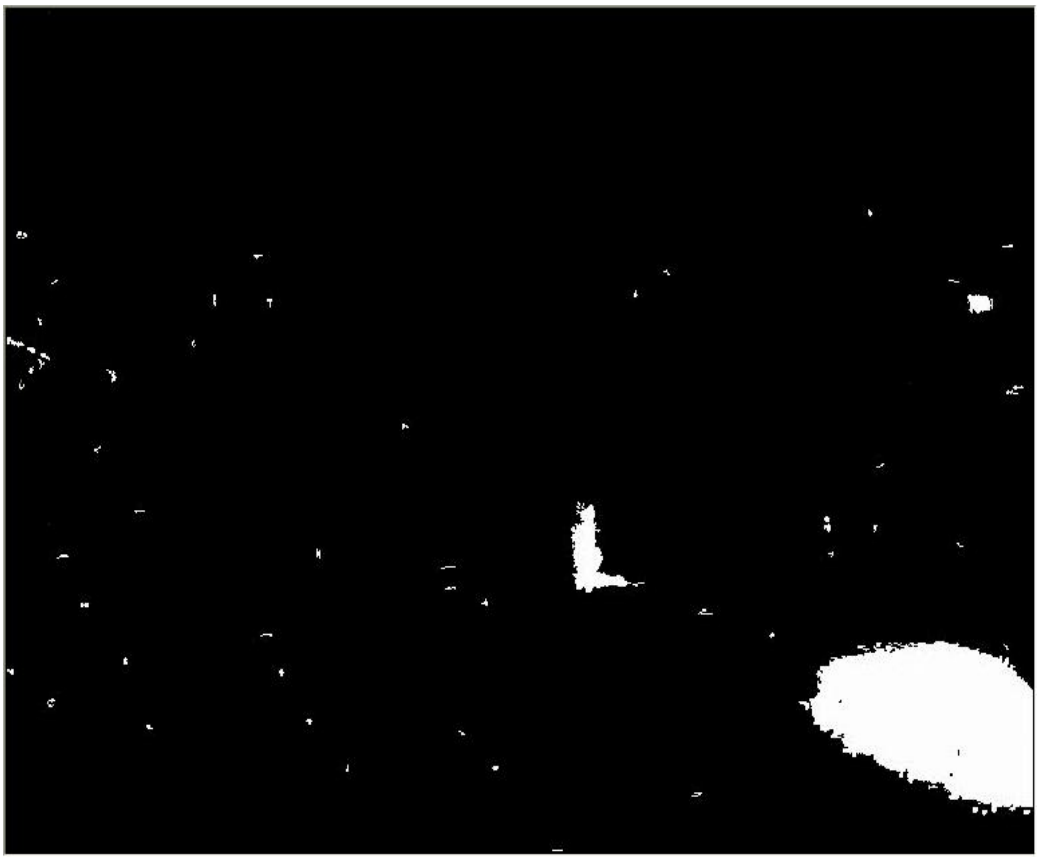
alpha2 = 0.01

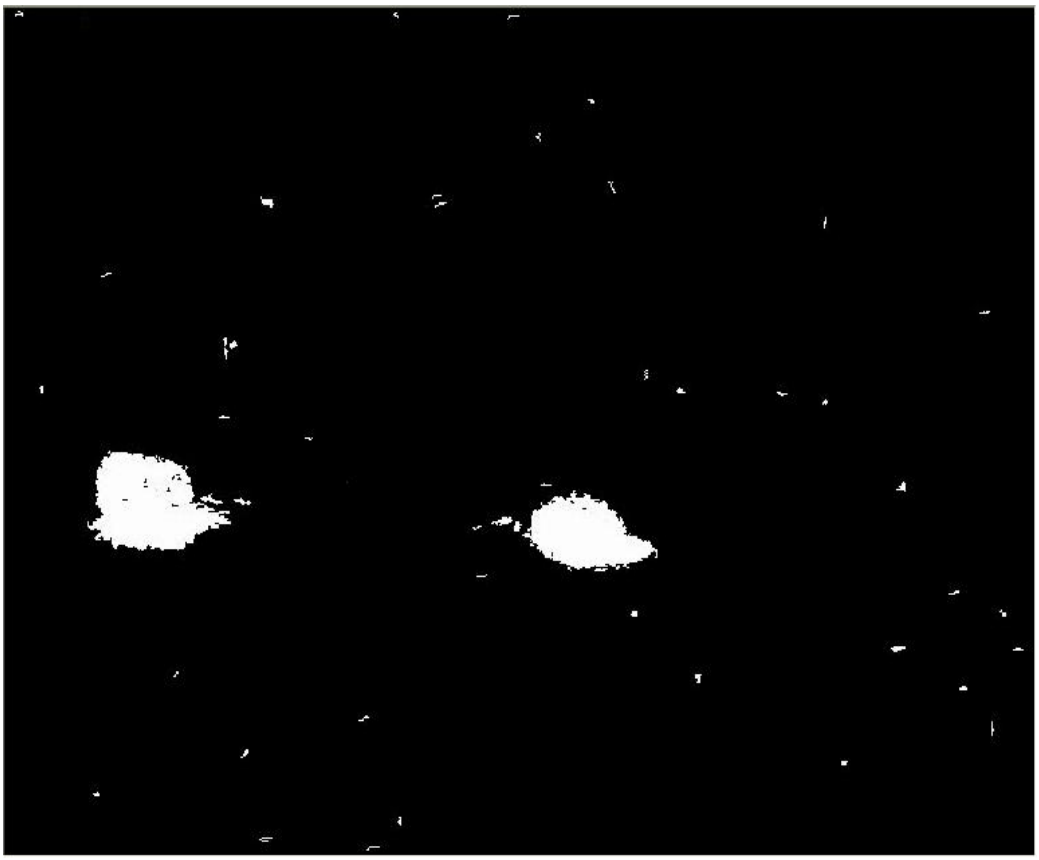




Para el primer valor de *alpha2*, 0.005, el vehículo es totalmente asimilado como background en el frame 0956, mientras que para el valor 0.01, esta circunstancia se produce en el frame 0782.

Con MOG se puede decir lo mismo que para el otro algoritmo respecto al comportamiento observado. A continuación se muestran las imágenes que ayuden a describir la situación que se produce.





En este caso, el vehículo es completamente asimilado como background en el frame 1008.

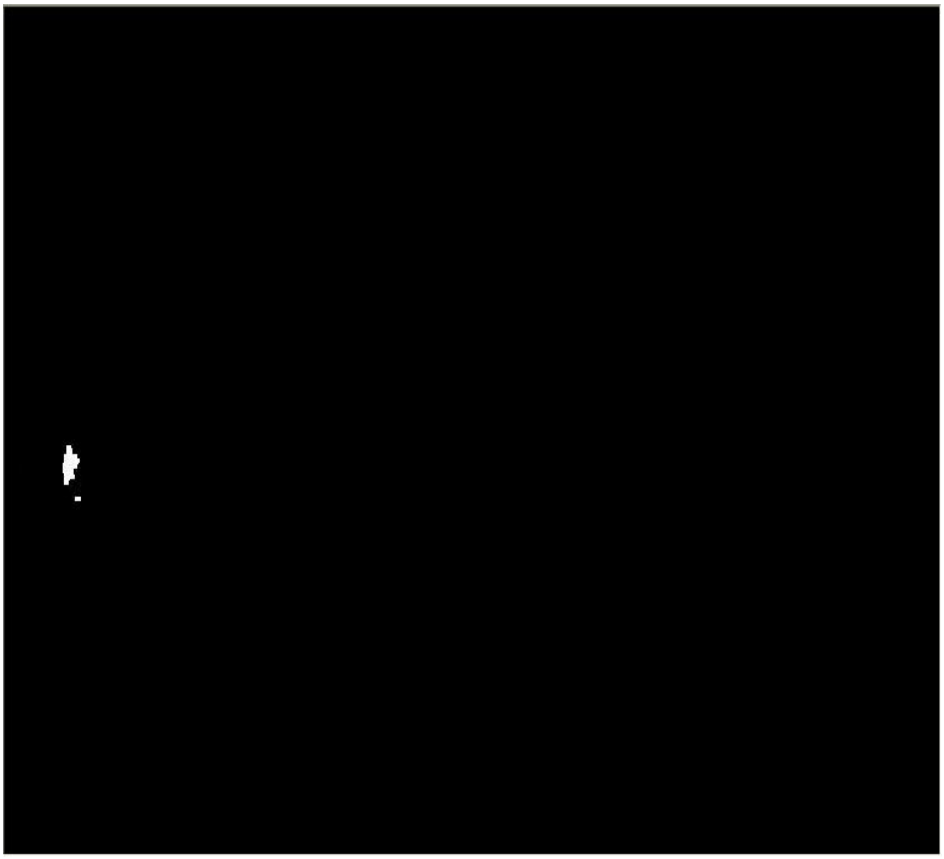
- *Background dinámico*: Se considera en este punto cualquier tipo de background dinámico, cualquiera de los vistos en la explicación de los algoritmos. Para ejemplificar, se tratan dos observados en la experimentación, y que afectan de manera diferente al comportamiento global del algoritmo:

    - o Movimiento de árboles: Habitual en escenarios de exterior cuando se observa la presencia de árboles cuyas ramas se mueven debido a la presencia de viento.

      Mostraremos el caso de un árbol, presente en el entorno del campus universitario, cuyas ramas se mueven cuando el viento sopla con fuerza. Éste se encuentra en la izquierda del vídeo, apreciándose solamente parte de sus ramas. Se puede decir que su incidencia no es de gran relevancia, puesto que la zona del background donde se produce esta situación es reducida. A continuación, mostramos la situación con dos imágenes. La primera de ellas muestra la situación inicial del árbol, y la segunda la nueva posición de las ramas debido al viento.

Si nos fijamos bien, se puede apreciar como en la segunda imagen las ramas del árbol se encuentran a una mayor altura en comparación con la primera imagen.

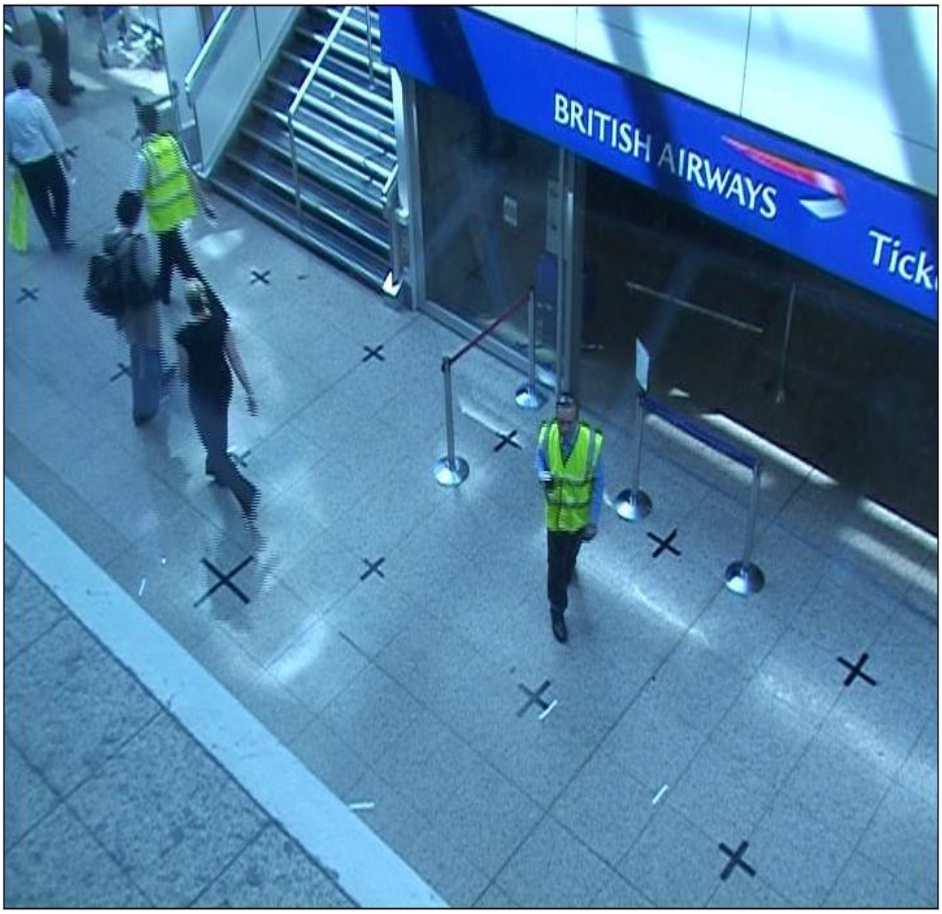
Con FGD se observa un buen comportamiento, ya que en ningún momento detecta a las ramas como foreground. A continuación mostramos las imágenes que constatan este comportamiento.



Con MOG, por el contrario, los movimientos de las ramas hacen que éstas sean detectadas en ocasiones como foreground. A continuación se muestran las imágenes que reflejan este comportamiento.

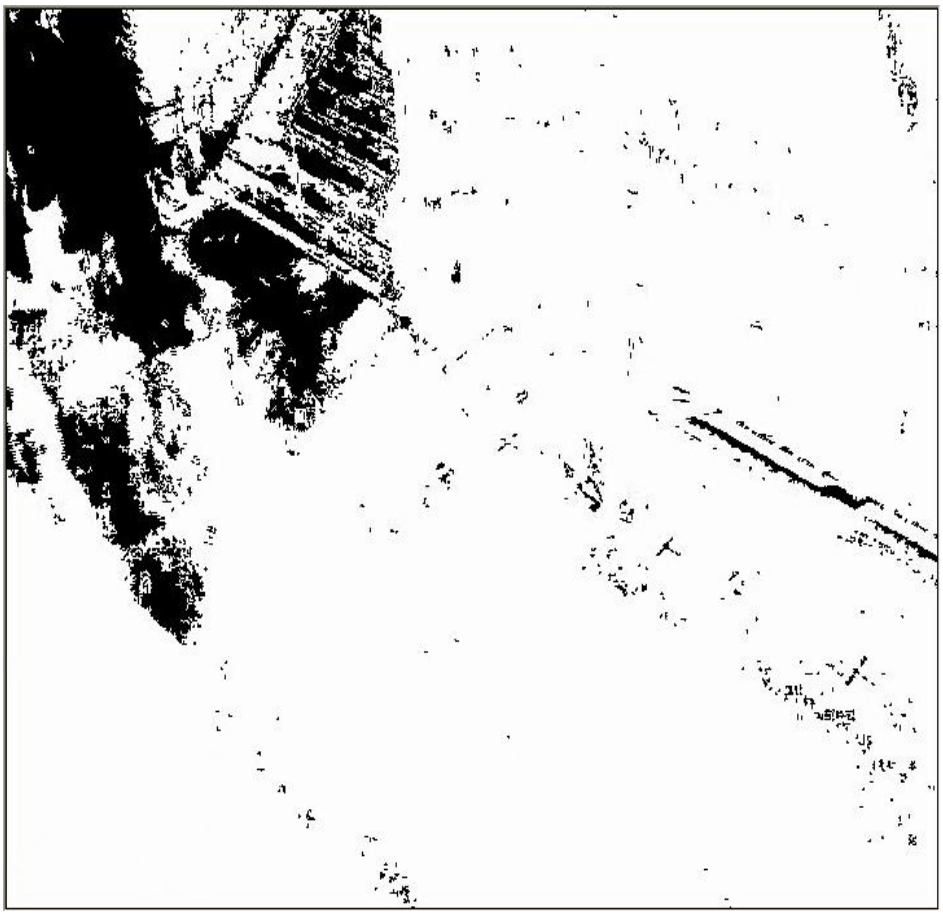
- Cambios de iluminación: Habitual en escenarios de exterior cuando zonas concretas del background varían su color debido a cambios de iluminación (por ejemplo, a causa de la aparición o desaparición de una nube que se interponga o no entre el sol y la superficie).

  Veremos un ejemplo de esto que se da en el último entorno visto, el del aeropuerto. En él veremos como gran parte de la superficie varía su color debido a que progresivamente se va oscureciendo la escena. Se puede decir que su incidencia es mayor que en el caso anterior, puesto que abarca una superficie muy significativa del background. A continuación veremos dos imágenes que expliquen esta situación. La primera de ellas refleja la situación inicial, y en la segunda se aprecia el nuevo color de la superficie motivado por el cambio de luminosidad.



Con FGD el comportamiento es el adecuado, ya que en todo momento sigue considerando a la superficie como background, es decir, como lo que realmente es. Vemos como se comporta el algoritmo cuando la superficie varía su color.

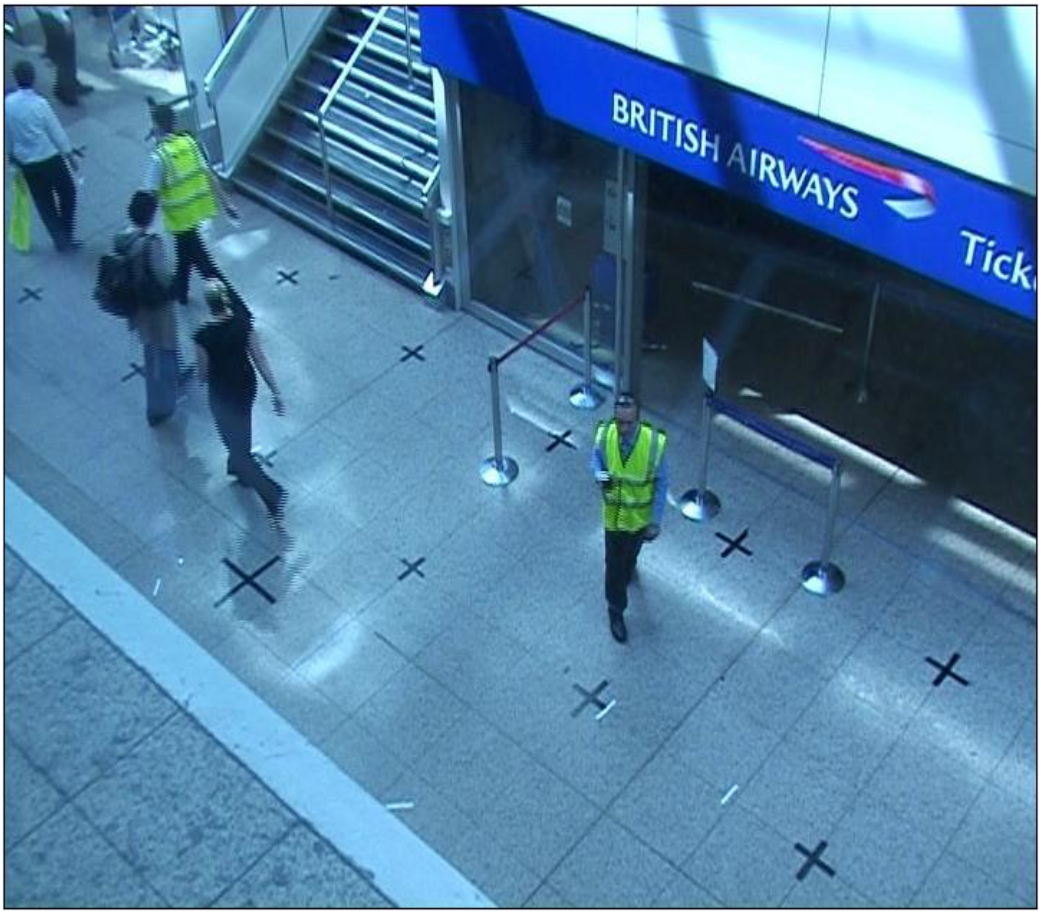
Con MOG detecta como foreground las zonas afectadas por el cambio de luminosidad. Vemos como se comporta el algoritmo cuando la superficie varía su color.



FOREGROUND:

- *Vídeo comienza con foreground en escena*: Hace referencia a vídeos que presentan objetos de foreground en la escena desde el primer frame del mismo.

  Parece interesante conocer cuál es el comportamiento que presentan nuestros algoritmos respecto a este punto, principalmente en lo referente al tiempo que tardan en detectar esos objetos. Sirva de ejemplo el vídeo utilizado en el punto anterior, donde inicialmente encontramos personas en la escena. Nos fijaremos, concretamente, en la persona situada en el centro de la imagen, con un chaleco reflectante. Volvemos a mostrar la imagen a las que nos estamos refiriendo.



Tanto con FGD como con MOG se observa un comportamiento similar. Al comenzar el vídeo se aprecia como la persona es ligeramente detectada (por partes), y al paso de poco tiempo la persona es detectada completamente. Para ambos algoritmos esa detección completa se produce en el frame 30, como se muestra en las dos siguientes imágenes, la primera referente al algoritmo FGD, y la segunda al algoritmo MOG.

Tras ejecutar nuestra aplicación, y ver cómo se comportaba, la impresión que había era que para este punto el valor del parámetro *Latency*, asociado al módulo *Blob Detector* **BD_CC**, afectaba de manera directa al tiempo que se tardaba en detectar al objeto. Por ello, a la hora de generar los resultados, se han probado diferentes valores para este parámetro: 10 (valor por defecto), 2 y 20, e incluso se probó con el otro módulo de *Blob Detector*, el **BD_Simple**, y se puede concluir que estas variaciones no afectan al comportamiento que se ha explicado anteriormente, es decir, el tiempo que tarda en detectar por completo a los objetos de foreground es independiente del valor del parámetro *Latency*.

- *Foreground "estático"*: Hace referencia a aquellos objetos de foreground que, por una razón u otra, se encuentran parados en un momento dado. Diferenciamos distintos casos:

    - Se paran pasado un tiempo: Incluimos en este punto a objetos en movimiento que se paran durante un largo período de tiempo.

      El comportamiento de nuestros algoritmos respecto a este punto es análogo a uno descrito anteriormente, el que explicaba la incorporación de

background a la escena. Por la tanto, mientras el objeto esté en movimiento será detectado como foreground. Una vez que el objeto se detenga, será mantenido como foreground durante un tiempo, tiempo que estará en función del algoritmo y parámetros utilizados, como ya se explicó antes. Cuando el objeto retome su movimiento, volverá a detectarse como foreground.

- Parado inicialmente: Incluimos en este punto a objetos de foreground que se encuentran parados en la escena en el primer frame del vídeo. Se pueden diferenciar varios casos:

  - Reanuda movimiento: En este caso, pasado un tiempo, el objeto en cuestión comienza a moverse.

    Consideremos el ejemplo de una persona que se encuentra parada en la escena al iniciarse el vídeo, y pasado un tiempo sale de la misma caminando. Esta situación ocurre en el entorno del centro comercial, y a continuación se muestra el primer frame del vídeo donde se puede observar lo que se comenta.

    Ambos algoritmos se comportan de igual manera frente a esta situación, esto es, detectando el objeto como foreground cuando comienza a moverse, y manteniendo durante un tiempo como foreground la zona ocupada inicialmente por el objeto parado. La experimentación con diferentes vídeos nos dice que ese tiempo dependerá del tiempo que permanezca el objeto parado, de esta forma, cuanto más tiempo esté parado el objeto más tiempo tardará en asimilar esa región como background. Se puede considerar como lógico este comportamiento, ya que cuando comienza el vídeo y se encuentra un objeto parado, éste es considerado como background, al igual que ocurre con el resto de background estático. Ahora bien, al

moverse ese objeto, la zona que éste ocupaba cambia de color, y por tanto, es lógico que la detecta como foreground hasta que el nuevo background sea aprendido.

Centrándonos en nuestro ejemplo, vemos que ese tiempo que tarda en asimilarse como background la zona ocupada por la persona es menor para el caso de MOG en el caso de que se utilicen los parámetros por defecto. Ahora bien, con FGD ese tiempo puede disminuirse aumentando el valor de *alpha2*. De esta forma, se ha probado que para el algoritmo MOG, es en el frame 114 cuando la región es asimilada totalmente como background, mientras que con el algoritmo FGD, haciendo uso de los valores para *alpha2* 0.005 (valor estándar), 0.01 y 0.1, los frames donde se produce esa situación son el 382, 269 y 102, respectivamente.

- No reanuda movimiento: En este caso, los objetos permanecen durante todo el vídeo en la misma región.

Veremos un ejemplo de este punto que se produce en el primer vídeo del primer entorno analizado. En él se puede observar una pequeña región donde se encuentran dos personas que realizan ligeros movimientos (movimientos análogos a los que puede realizar una persona que se encuentra atendiendo en una tienda a los clientes). Mostramos, a continuación, una imagen que muestra la situación descrita.

Con FGD no llega a detectar con facilidad este tipo de foreground, su comportamiento es parecido al que tiene con el background dinámico. Si además, estos objetos apenas se ven (por ejemplo: que estén en una zona oscura y apenas haya contraste entre ellos y el background), los problemas se acentúan. Mostramos una imagen correspondiente al mismo frame anterior, donde se refleja esta situación.

Con MOG detecta mejor los movimientos que puedan realizar esas personas, aunque no sean muy apreciables. Mostramos una imagen correspondiente al mismo frame, donde se refleja esta situación.

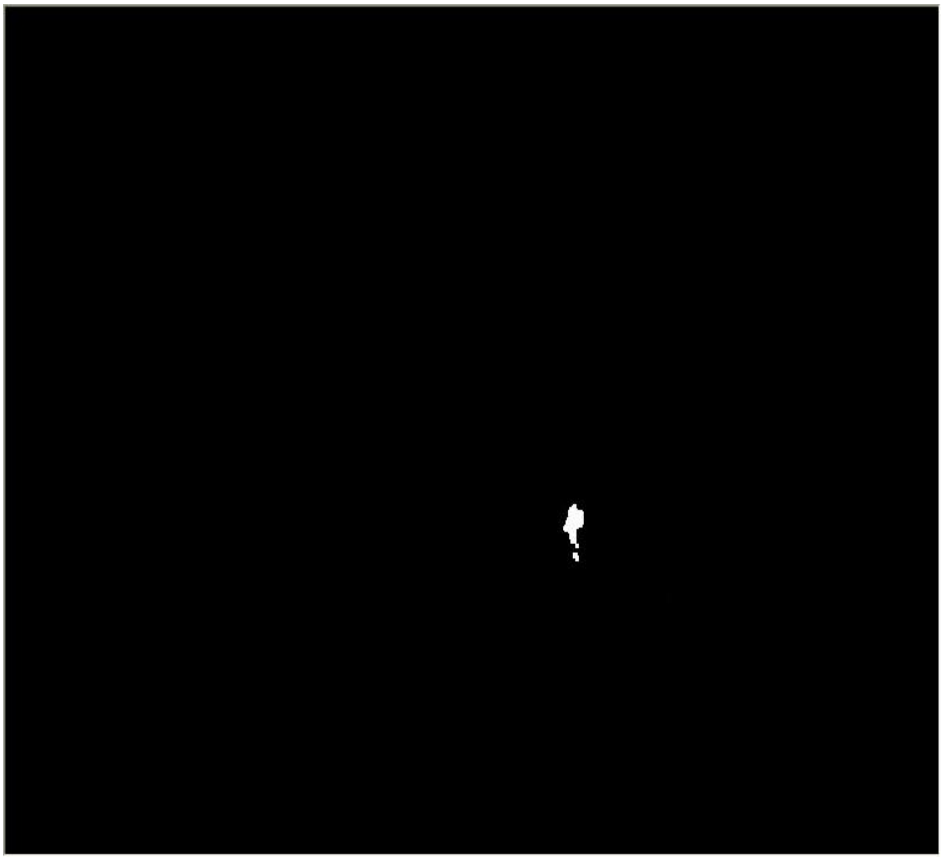
- *Foreground "dinámico" y solitario*: Se considera en este punto a todo aquel objeto de foreground que se desplaza y lo hace en solitario. Tenemos en cuenta las siguientes dos variantes:

    - Sin paradas: En este caso, el objeto de foreground no realiza ningún tipo de parada durante el tiempo que aparece en la escena.

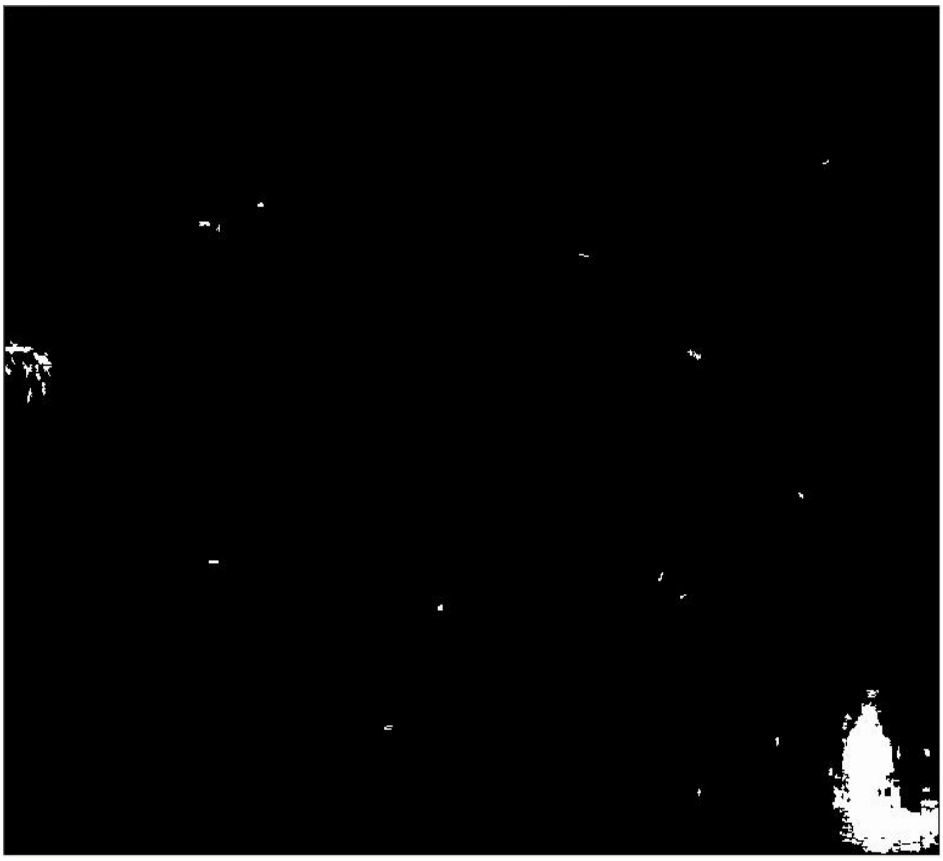
      Tras la etapa de experimentación, se llegó a la conclusión de que es de gran relevancia la distancia a la que se encuentre el objeto de la cámara. Por tanto, veremos que el resultado de nuestros algoritmos estará influenciado por ese factor. Lo vemos mejor en un ejemplo, donde el resultado que se muestra es aplicable a todas las situaciones en las que se presente este tipo de foreground.

      Consideramos un vídeo grabado en el campus universitario. En él se puede apreciar cómo aparece una persona en la escena por la zona más cercana a la cámara, y poco a poco se va alejando hacia un vehículo. A continuación, mostramos dos imágenes que nos van a servir de ayuda para explicar este punto, la primera será un ejemplo de objeto de foreground cercano a la cámara, y la segunda de un objeto más alejado a la cámara.

El comportamiento de FGD se ve afectado por la distancia de la persona a la cámara. La detección cuando la persona está cercana a la cámara se puede considerar muy buena, ya que la detecta de manera muy “perfecta”, es decir, que representa de manera muy real la figura de la persona. Sin embargo, cuando la persona se va alejando, la detección va siendo cada vez más pobre. A continuación mostramos dos imágenes obtenidas del resultado de FGD, y que se corresponden con las dos imágenes expuestas anteriormente.



El comportamiento de MOG no se ve afectado por la distancia de la persona a la cámara. Sin embargo, la detección no es tan “perfecta” como vimos en el otro algoritmo. A continuación mostramos dos imágenes obtenidas del resultado de MOG, y que se corresponden con las dos imágenes expuestas anteriormente.

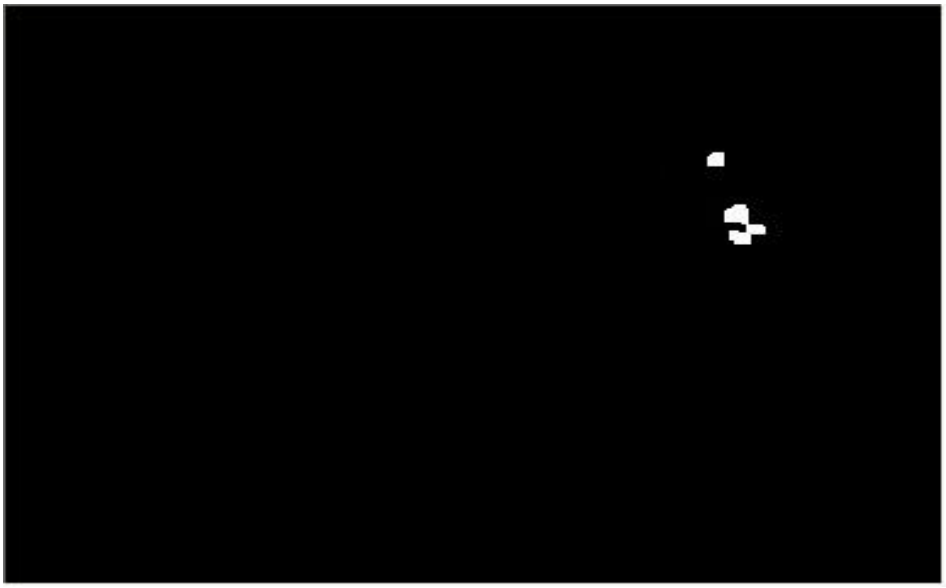


- Con paradas: En este caso, el objeto de foreground realiza alguna parada durante el tiempo que aparece en la escena. Eso sí, se trata siempre de paradas no muy prolongadas.

Con ambos algoritmos se observa un comportamiento muy similar, esto es, que, al tratarse de paradas cortas, no se llega a perder la detección del objeto como ocurría con las paradas de mayor duración. Eso sí, hay que tener en cuenta que este comportamiento no es siempre así, ya que, en ocasiones, cuando el color del objeto de foreground no contrasta mucho con el color de background que ocupa, la detección con el algoritmo FGD presenta dificultades, perdiendo durante la parada las partes del objeto cuyo color se parece con el del background. A continuación, mostramos un ejemplo de esta situación.

Consideramos el primer vídeo del primer entorno. En él, hay una persona que realiza una parada corta. En la siguiente imagen mostramos el momento en el que la persona se detiene.

Con FGD, vemos la complicación que se describía anteriormente. Como se puede ver en la imagen anterior, la camisa del hombre tiene un color muy parecido al del suelo. Esta circunstancia afecta al resultado de este algoritmo, ya que, como se muestra en la siguiente imagen, se pierde la detección de la camisa.

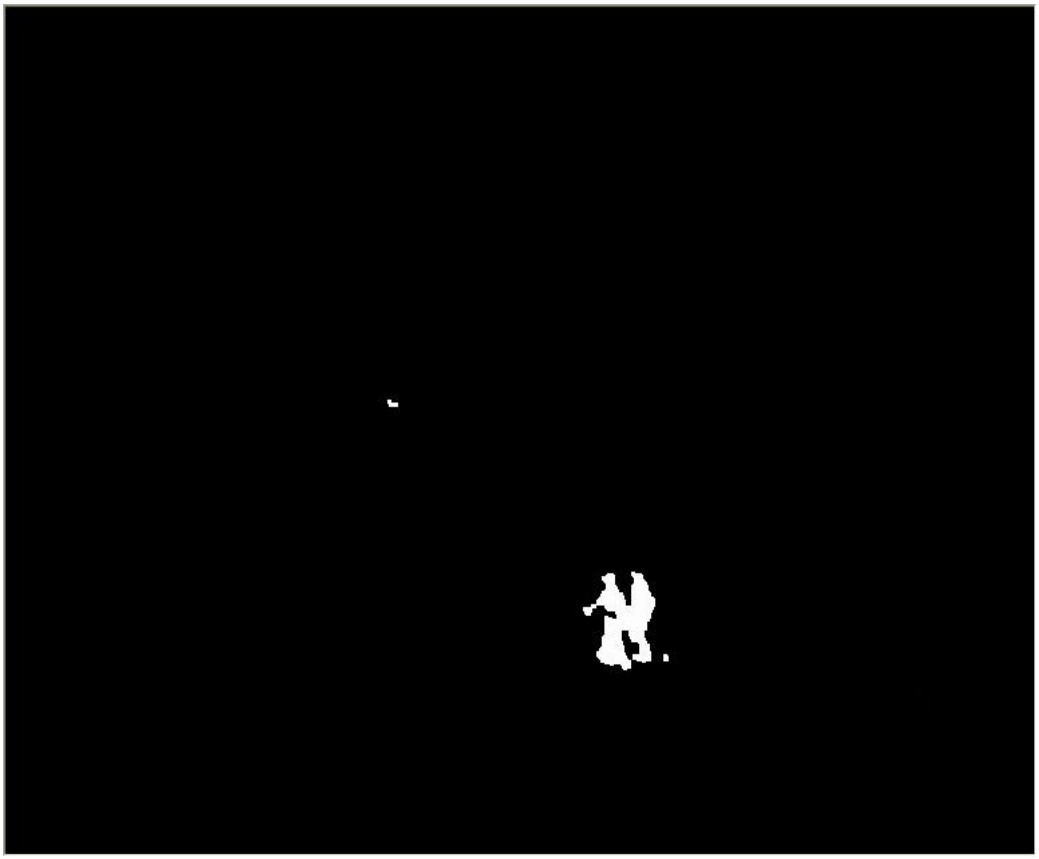


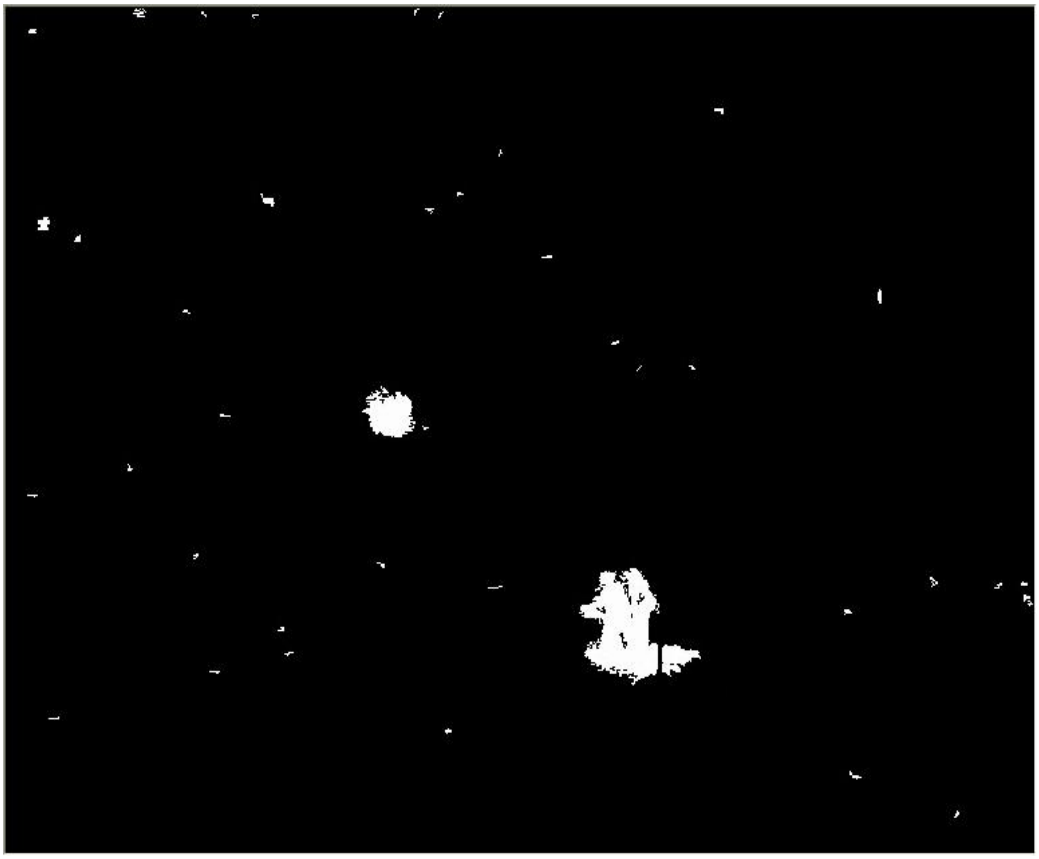
Con MOG, por el contrario, el resultado no se ve afectado por tal circunstancia. A continuación se muestra una imagen del resultado del mismo frame con este algoritmo.

- *Foreground "dinámico" y en grupo*: Consideramos en este punto al conjunto de dos o más personas (u objetos de foreground cualesquiera) que se mueven muy cercanas las unas a las otras.

  Un ejemplo de este punto es el que describimos a continuación. En un vídeo del entorno del campus universitario, podemos apreciar, en un momento dado, como salen a escena dos personas caminando muy pegadas. La siguiente imagen nos muestra tal situación.

Con ambos algoritmos se pude observar un correcto comportamiento, ya que diferencia bien a las personas que van caminando juntas. Este comportamiento se ha repetido en los diferentes vídeos sobre los que se ha experimentado. A continuación, se muestran dos imágenes referentes a los resultados obtenidos con el algoritmo FGD y MOG, respectivamente.

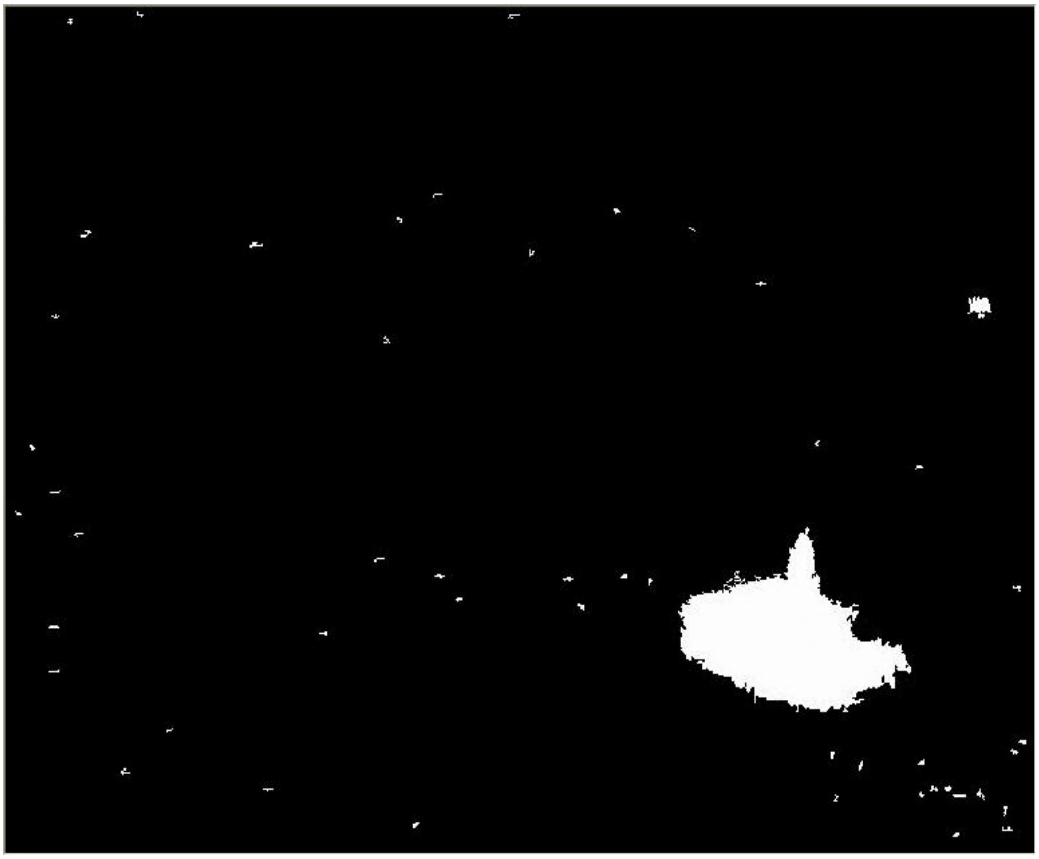




- *Foreground "dinámico" que se cruza*: Consideramos en este punto a objetos de foreground que se mueven y llegado un momento se cruzan con otros.

Para ejemplificar esta situación, volveremos a hacer uso del vídeo utilizado en el punto anterior. En este vídeo, en un momento dado, se puede observar como se produce un cruce (o solapamiento) entre una persona que va caminando y un vehículo que circula en sentido contrario a la persona. La siguiente imagen muestra esta circunstancia.

Con ambos algoritmos vuelve a obtenerse un comportamiento correcto. Al igual que ocurría en el punto anterior, el cruce no afecta a que se siga manteniendo una buena detección de los objetos que se cruzan. Vemos en las dos siguientes imágenes el resultado de esta situación para los algoritmos FGD y MOG, respectivamente.



- *Foreground con sombras*: Consideramos en este punto a los objetos de foreground que van “acompañados” de sus sombras.

A lo largo de todos los ejemplos que hemos ido analizando, se ha podido ver cuál es el comportamiento que tienen nuestros algoritmos frente a este punto. Si nos fijamos, en los resultados de FGD no se considera foreground a las sombras de los objetos, mientras que con el algoritmo MOG tales sombras son consideradas, al igual que el objeto al que pertenecen, como foreground. Como ya se ha dicho, este comportamiento se puede observar en los anteriores ejemplos expuestos, si bien cabe recalcar que este comportamiento es el general, el más usual.

Sin embargo, hay ocasiones en los que el color de las sombras de los objetos de foreground contrasta mucho con el color del background. Es en estos casos cuando el resultado de FGD puede variar su comportamiento general, detectando esas sombras como foreground. Mostramos a continuación ejemplo que prueba esto.

Se trata de un vídeo grabado en el centro comercial. Se observa a una persona caminando, cuya sombra es más apreciable al ser el suelo de color más claro. La siguiente imagen muestra esta situación.

El resultado de FGD en este frame es el mostrado en la siguiente imagen.

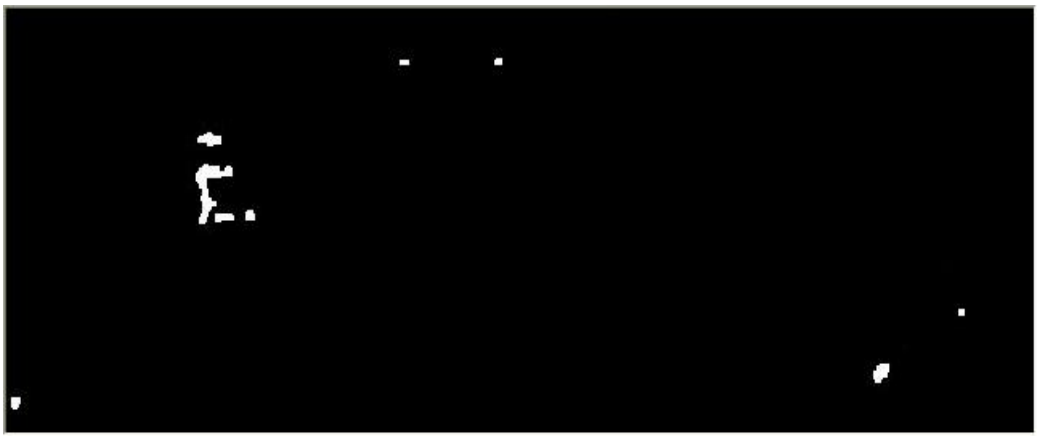
- *Foreground reflejado*: Consideramos en este punto a todo objeto de foreground que va "acompañado" de su reflejo, ya sea en cristales, espejos…

Durante la experimentación nos encontramos con ejemplos de este punto en el entorno del centro comercial, donde en un escaparate se reflejaba el interior de la tienda, o en el entorno de la estación de tren, donde había una especie de cristaleras en las que se reflejaban las personas que caminaban por un pasillo.

Respecto al comportamiento que tienen nuestros algoritmos sobre este punto, se puede asemejar con el comentado en el punto anterior. En ambos casos, tanto sombras como reflejos, son generalmente poco apreciables, por lo que el algoritmo MOG tiene más facilidad de detectarlos como foreground, mientras que el algoritmo FGD solamente los considera como objetos de foreground cuando son muy visibles.

A continuación, presentamos un ejemplo de esto. Sucede en el mismo vídeo tratado en el punto anterior, en el centro comercial, donde al poco de iniciarse se observa en el escaparate un reflejo de una persona que se encuentra dentro de la tienda. La siguiente imagen captura ese instante.

El resultado de FGD en este frame es el mostrado en la siguiente imagen.



4. EVALUACIÓN

En esta sección se verán diferentes herramientas que existen para evaluar los algoritmos de segmentación.

Aunque no se comentó en su momento, la mayoría de los vídeos que se expusieron en la etapa de experimentación, exactamente todos menos los relativos al primer entorno, son vídeos generados de datasets propios de **PETS** (**P**erformance **E**valuation of **T**racking and **S**urveillance). Por este motivo, parece buena idea fijarnos en cómo trata PETS la evaluación de los algoritmos. Por lo tanto, nos centraremos en el servicio de evaluación on-line que proporciona PETS, donde se incluyen una serie de métricas utilizadas para evaluar los resultados de los algoritmos.

4.1. Métricas PETS: Servicio de Evaluación de Funcionamiento On-Line

A modo de resumen, podemos comentar que en este apartado presentaremos el Servicio de Evaluación On-Line de Métricas PETS para algoritmos de vigilancia de computación visual. Este servicio permite a los investigadores enviar los resultados de sus algoritmos para su posterior evaluación frente a un conjunto de métricas aplicables. Los resultados de los procesos de evaluación son mostrados públicamente permitiendo a los investigadores ver instantáneamente como funcionan sus algoritmos frente a otros con el mismo propósito enviados previamente.

Introducción.-

La evaluación del funcionamiento de los algoritmos de segmentación es algo necesario para poder concluir si la comunidad de investigación está haciendo una progresión cuantificable en el desarrollo de algoritmos. Previamente hubo una tendencia a probar y comunicar los resultados del algoritmo basados en datasets propios. Esto podría inducir a unas conclusiones no muy precisas debido al uso de unos datasets no representativos de un problema particular. Para combatir este problema se crearon datasets estándar: PETS, CAVIAR, ETISEO e iLIDS. Incluso con datasets estándar puede ser complicado confirmar el funcionamiento de unos algoritmos debido a evaluaciones con métricas internas y comunicación selectiva de resultados. El siguiente paso progresivo, por tanto, es proporcionar a la comunidad de investigación una ubicación central donde los algoritmos

puedan ser probados con datasets estándar y un conjunto común de métricas. El sitio web de las Métricas PETS tiene la intención de proporcionar este servicio para la comunidad de investigación.

Las Métricas PETS fueron desarrolladas para ser un mecanismo tanto auxiliar como complementario de los tradicionales talleres de PETS, donde se discutían asuntos relativos a la evaluación de algoritmos. El objetivo final es proporcionar un mecanismo automático para comparar, de una manera cuantitativa, un conjunto de algoritmos operando con los mismos datos. Métricas PETS automatiza el proceso de evaluación de funcionamiento, y proporciona a la comunidad un depósito en línea de datasets, métricas y resultados. El acercamiento es diferente de actividades relacionadas, como ETISEO, donde el coordinador global desarrolla la evaluación de resultados enviados. En Métricas PETS, los resultados son cargados directamente al sitio web, automáticamente evaluados y presentando los resultados clasificados junto a otros algoritmos.

Una principal motivación detrás de Métricas PETS estuvo en la evaluación de resultados presentados durante PETS 2001. En este segundo taller de PETS, se puso el requisito o exigencia de que los papeles enviados fueran acompañados con la salida de resultados del algoritmo en formato XML. El coordinador entonces reconstruyó la detección de objeto y resultados de rastreo de archivos XML, que previenen una comparación cualitativa de un número de algoritmos operando en las mismas secuencias de video PETS. Un resultado significativo de este proceso fue que esto permitió a subproblemas específicos dentro de la tarea de vigilancia, como por ejemplo, el aseguramiento del mantenimiento de la identidad de objetos rastreados por oclusión parcial, ser estudiados. Específicamente, qué algoritmos triunfaron y fallaron en esta tarea para una secuencia dada. Esto condujo a recomendaciones de cómo combinar las ventajas de diferentes algoritmos para producir un simple y más robusto algoritmo de rastreo. Un objetivo importante de Métricas PETS es extender la metodología de evaluación a:

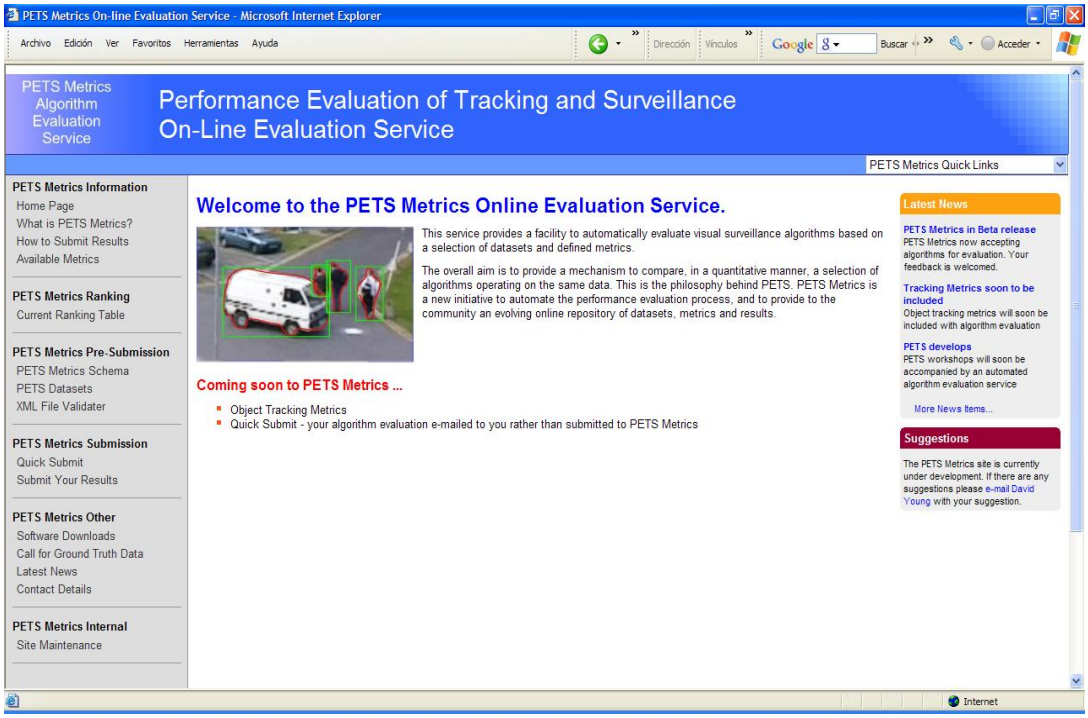
1. Proporcionar un depósito en-línea de datasets, métricas y resultados.
2. Tener en cuenta la evaluación *automática* de resultados enviados.
3. Proporcionar resultados cuantitativos que pueden ser vistos clasificados por métrica.

Sitio Web de Métricas PETS.-

Descripción del Sitio Web.-

El sitio web de Métricas PETS es la interfaz utilizada por los investigadores para enviar los resultados de sus algoritmos y ver el funcionamiento de los mismos frente a un conjunto

de métricas aplicables. Para cada métrica, el sitio web muestra periódicamente una tabla del ranking que refleja cómo los algoritmos enviados han funcionado frente a los otros. A continuación, mostramos una imagen del sitio web, a la que podemos acceder en la siguiente URL http://www.cvg.cs.rdg.ac.uk/cgi-bin/PETSMETRICS/page.cgi?home.



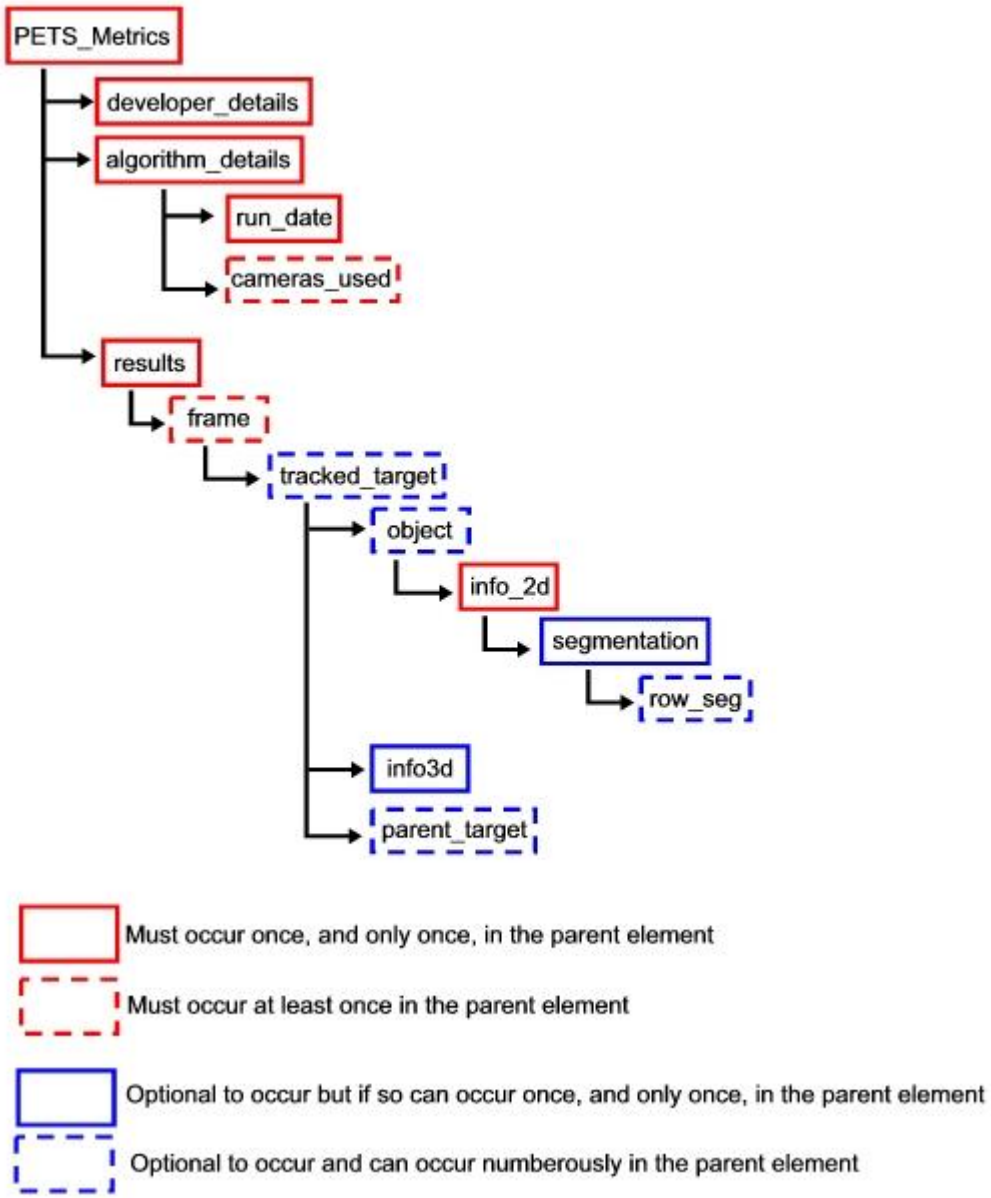
El sitio es escalable para acomodar cualquier número de métricas para cualquiera de las áreas o actividades particulares de investigación. El sitio actualmente incluye, pero no está limitado a ello, métricas para segmentación de movimiento. En el futuro las métricas estarán extendidas para cubrir tareas adicionales de investigación de vigilancia visual, por ejemplo, rastreo y clasificación.

El sitio es también escalable para cualquier número de datasets de vídeo. Debido a la naturaleza intensiva del tiempo de los ground truth de Métricas PETS, actualmente se evalúan algoritmos frente al Dataset 1 de la Cámara 1 de PETS 2001. Esto será expandido para cubrir la totalidad de los dataset de PETS 2001 y más datasets con ground truth disponible.

Archivos Resultados de los Algoritmos.-

Generalmente, los investigadores desarrollan sus algoritmos en diferentes plataformas de computación, usan una gran variedad de lenguajes de programación y típicamente almacenan sus resultados del algoritmo en sus propias estructuras de datos. Para Métricas PETS, para poder evaluar los algoritmos de los investigadores, debe existir un formato de archivo para el envío de los resultados. Para resolver esta cuestión, Métricas PETS

requiere que un archivo del resultado del algoritmo sea enviado en un archivo XML formateado al Esquema XML de Métricas PETS. Un Esquema XML es una definición sobre cómo construir legalmente un archivo XML. La siguiente imagen muestra esquemáticamente los elementos XML usados en el Esquema XML de Métricas PETS.



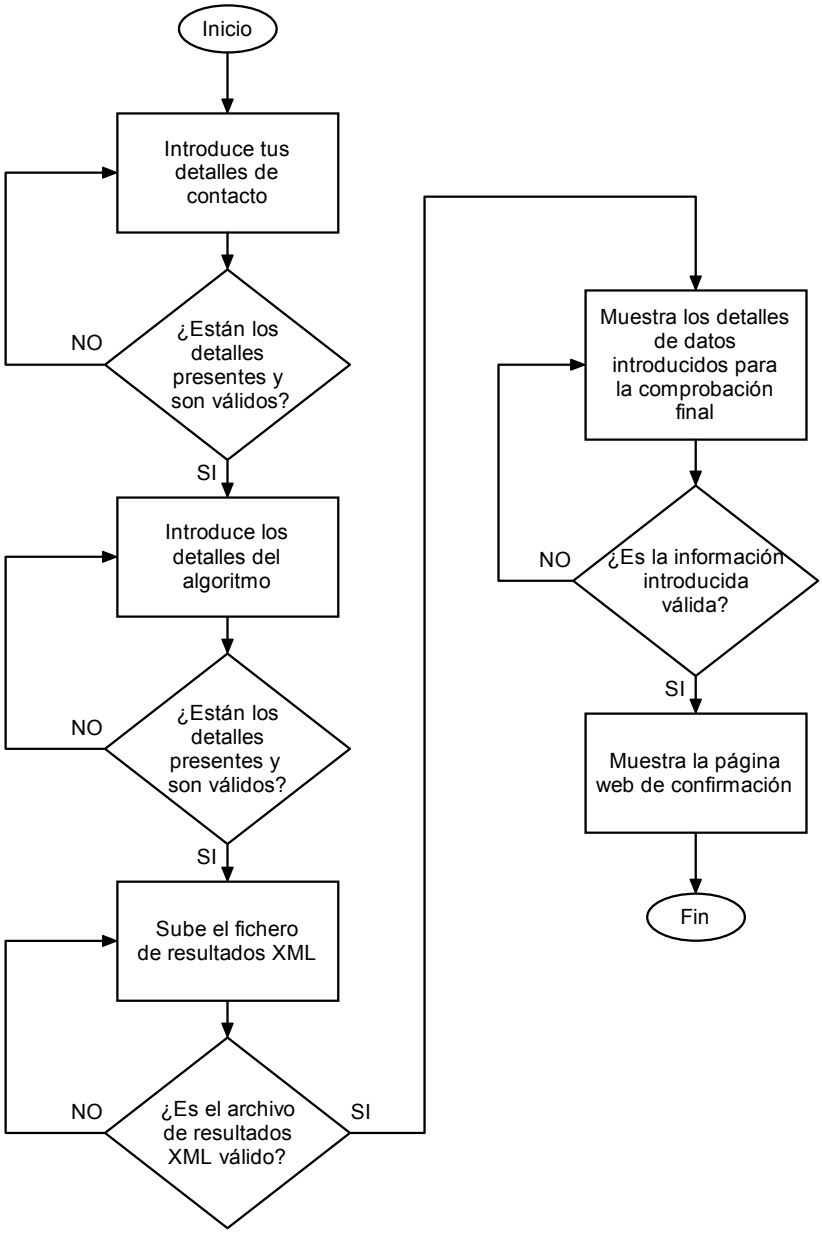
XML es un formato adecuado para el envío de resultados de Métricas PETS, ya que se trata de una plataforma de no propiedad, reconocida internacionalmente, y lenguaje de programación independiente del formato de los datos. Además, XML puede ser fácilmente generado por un fichero normal de lenguaje de programación escribiendo funciones propias del lenguaje.

Proceso de envío.-

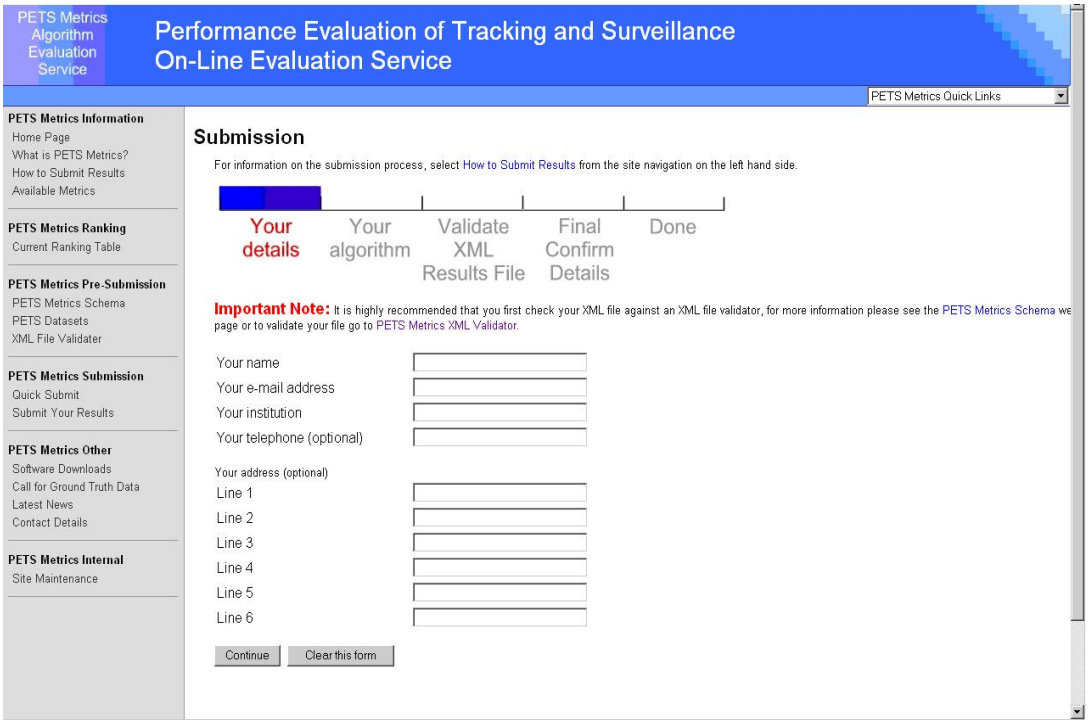
El envío de los resultados del algoritmo al sitio web de Métricas PETS consiste en las siguientes cinco etapas:

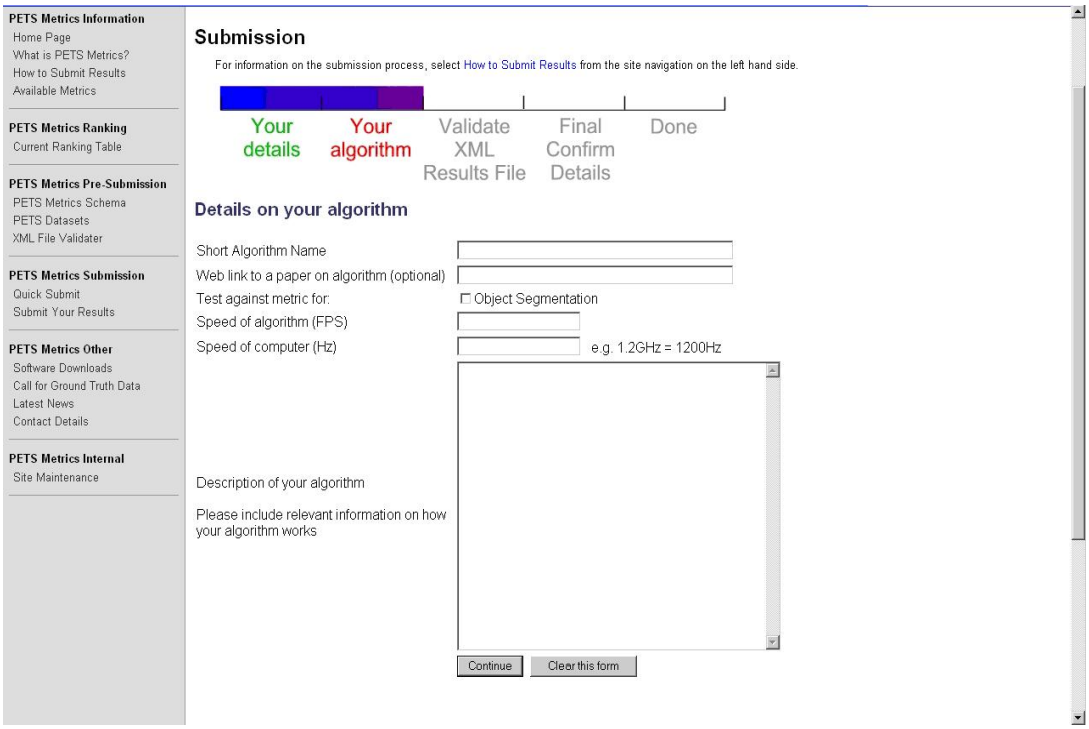
1. Introducción de los detalles de contacto.
2. Introducción de los detalles sobre el algoritmo.
3. Carga del archivo de Resultados XML.
4. Comprobación final de los datos introducidos.
5. Confirmación del envío.

Este proceso es resumido a continuación con el siguiente diagrama de flujo generalizado:

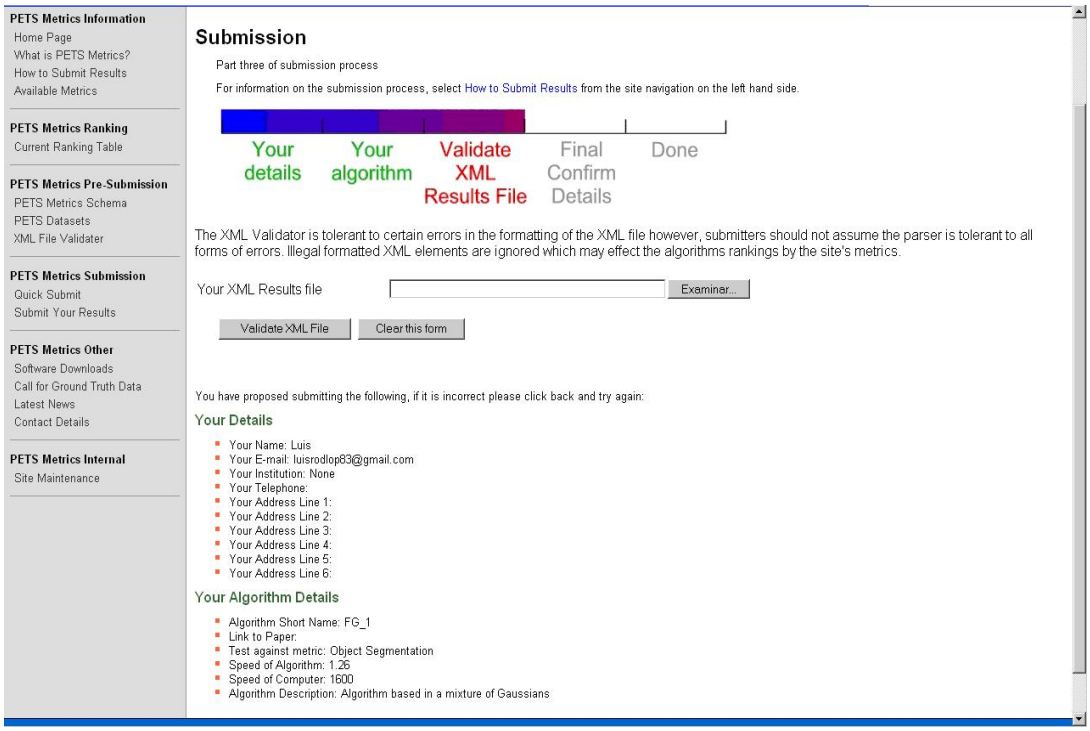


En primer lugar, el emisor completará dos cuestionarios vía web como primer paso para enviar sus resultados del algoritmo al sitio. El primer cuestionario captura los detalles de contacto del emisor para permitir futura correspondencia, mientras que el segundo cuestionario captura detalles relativos al algoritmo; éstos incluye: un nombre corto, frames por segundo (FPS) del algoritmo, velocidad de la ejecución del ordenador, descripción del algoritmo y tarea para la que el algoritmo fue desarrollado (una selección de una lista). Es requerido conocer para qué tarea fue desarrollado el algoritmo para aplicar las métricas correctas al fichero de resultados. Los FPS y la velocidad del ordenador son requeridos para analizar la carga de tratamiento aproximada del algoritmo. A continuación, mostramos los dos cuestionarios.

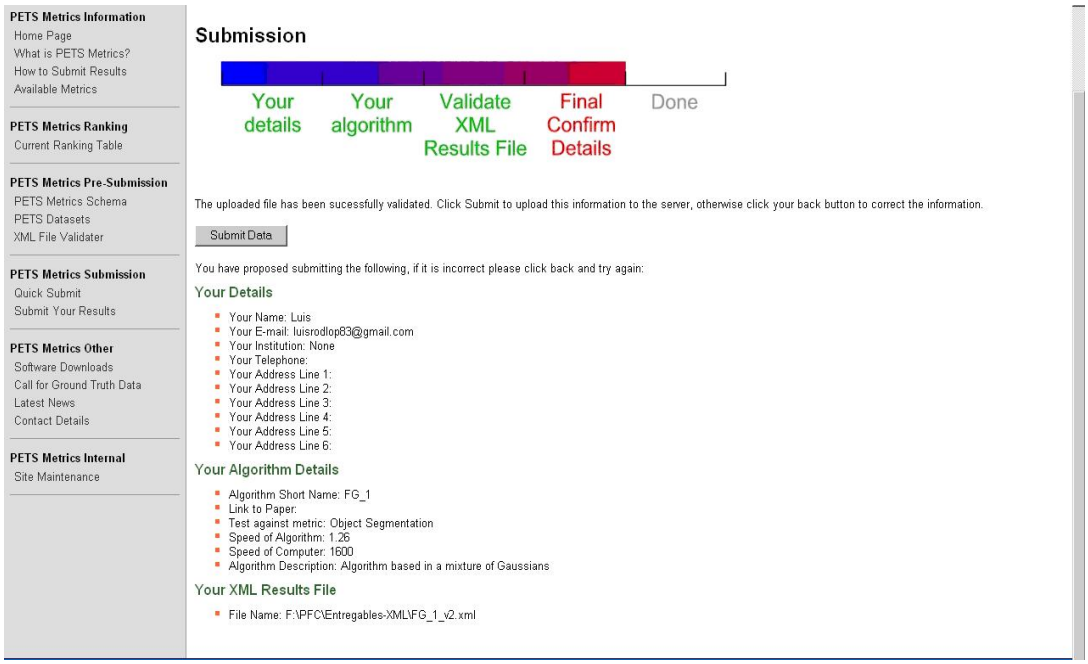


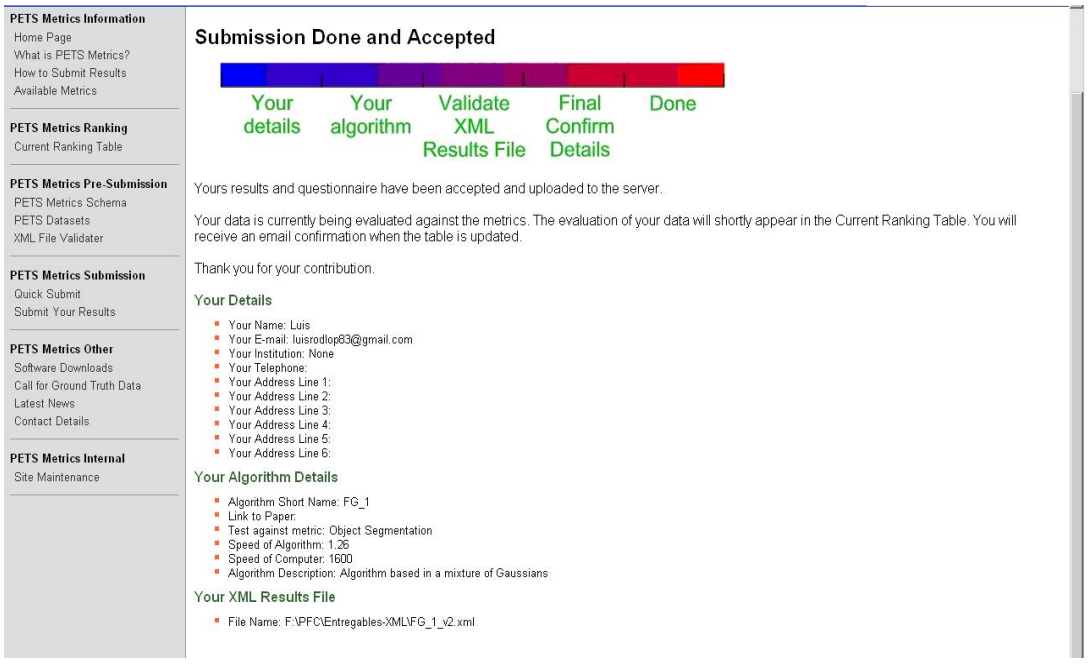


Para cargar un archivo a Métricas PETS el usuario selecciona un fichero desde su ordenador. El sito de Métricas PETS validará el archivo XML. El analizador gramatical de XML de Métricas PETS intentará extraer toda la información válida. Por flexibilidad, el analizador es tolerante a ciertos errores en el formato del archivo XML. Los elementos XML formateados ilegalmente son ignorados, ya que pueden afectar al ranking de los algoritmos. Esta etapa es reflejada en la siguiente imagen.



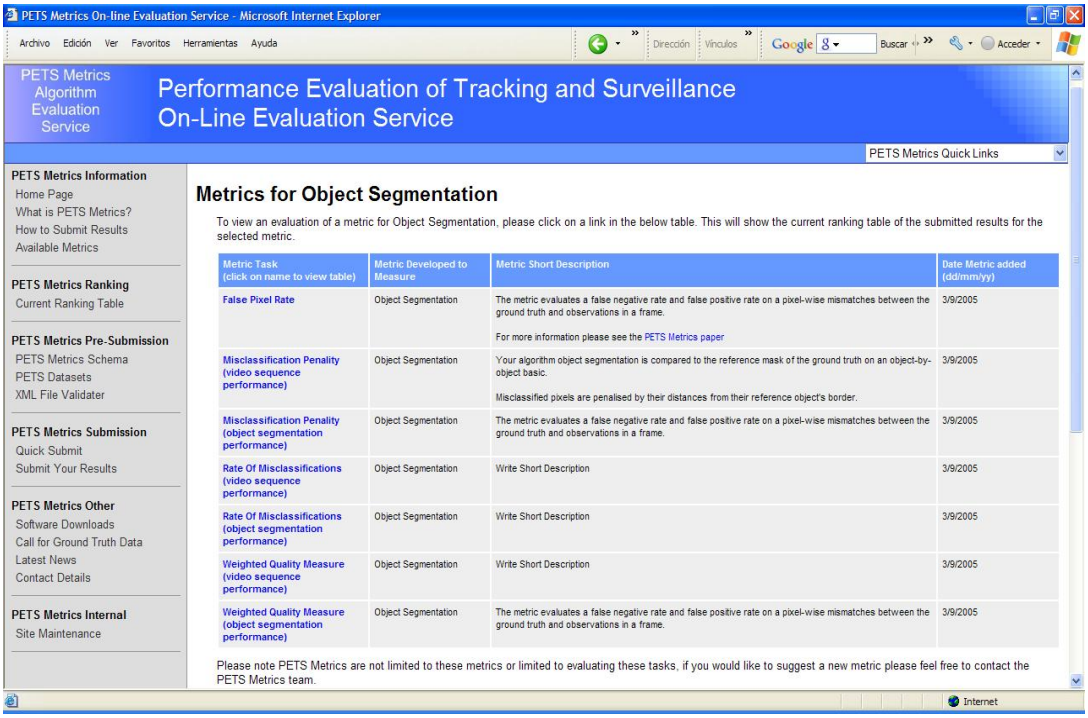
Antes de la confirmación de un envío satisfactorio a Métricas PETS, el emisor tiene una oportunidad final para corregir cualquier entrada del cuestionario. En la confirmación por parte del emisor de que el información del cuestionario es correcta, el sitio web de Métricas PETS muestra una página web de envío satisfactorio y comienza a procesar los resultados frente las métricas apropiadas. Estas dos últimas etapas correspondientes al proceso de envío de los resultados del algoritmo son reflejadas en las dos siguientes imágenes.





Despliegue del Ranking de Algoritmos.-

Mediante la selección de la opción Tabla del Ranking del sitio web de Métricas PETS, un usuario puede ver una tabla de las métricas actuales con su descripción asociada. Esta situación la reflejamos en la siguiente imagen.



Mediante la selección de una métrica, aparecerá una página web mostrando la tabla del ranking de los algoritmos para esa métrica, como se muestra en la siguiente imagen cuando seleccionamos la primera de las métricas.

Ranking table for the False Pixel Rate Metric

Metric Description

For detailed information regarding this metric please see the PETS Metrics paper

Ranking Table for all XML file results

Please click on a column title to re-order the table to that measure.
If this is not what you are expecting try refreshing your browser.

Result File (click for descriptions)	False Negative Rate Ascending Decending	False Positive Rate Ascending Decending	False Rate (overall evalation score) Ascending Decending
example.xml	0.00000	0.00000	0.00000
pets_metrics.MD_WLF.xml	0.10900	0.00143	0.05522
pets_metrics.MD_KDE.xml	0.15338	0.00122	0.07730
pets_metrics.MD_GMM.xml	0.16756	0.00131	0.08443
pets_metrics.MD_EDG.xml	0.20253	0.00082	0.10167
pets_metrics.MD_VAR.xml	0.20731	0.00085	0.10408
pets_metrics.MD_BC.xml	0.22002	0.00076	0.11039
pets_metrics.MD_DIF.xml	0.28953	0.00158	0.14555

Como se puede observar, una métrica particular puede tener muchas partes constituyentes, por ejemplo un valor por positivos verdaderos, positivos falsos, etc. Estas partes constituyentes serán mostradas como cabeceras de columna en la tabla ranking para esa métrica. Los algoritmos en la tabla de ranking para esa métrica pueden ser clasificados en diferente orden dependiendo de la parte constituyente seleccionada.

Para complementar la información en la tabla de ranking, una segunda tabla de información relativa a los algoritmos es también mostrada. La información de la segunda tabla proporciona los detalles del algoritmo capturados en el segundo cuestionario, explicado anteriormente. Esto permitirá a los usuarios de Métricas PETS identificar qué métodos de visión computacional están funcionando mejor.

4.2. Métricas PETS.-

Actualmente, las Métricas PETS evalúan algoritmos de segmentación de movimiento. Concretamente, 240 objetos de foreground de 100 frames de la secuencia de test de la Cámara 1 del Dataset 1 de PETS 2001 han sido “ground truthed”.

Ground Truthing.-

Las métricas evalúan algoritmos frente a un ground truth que es asumido que es correcto. Generar datos de ground truth es un proceso repetitivo y manual, donde el semi-automatismo debería ser únicamente introducido donde no se influya sobre los datos de ground truth.

En la práctica, la colección de datos de ground truth está sujeta al error subjetivo sistemático derivado de personas que se encuentran recopilando los datos. Por ejemplo, en las fronteras de un objeto de ground truthing, la cuantización del mundo real a píxeles y el ruido de JPG puede hacer difícil definir exactamente una frontera. En este caso el ruido sistemático es añadido al ground truth, comparando individuos, como alguno puede sistemáticamente tender a definir regularmente el límite del objeto o más lejos del objeto.

Idealmente, una secuencia dada debería ser ground truthed un número de veces por la misma persona (o por diferentes personas) y el resultado promediado. Esto es, sin embargo, normalmente irreal debido a los requerimientos económicos y de tiempo. Es, por lo tanto, prácticamente imposible declarar que los datos de ground truth colectados serán en realidad 100% correctos. Esto, en consecuencia, siempre conducirá a un nivel de error en los resultados relatados de métricas frente a los resultados de un algoritmo.

Asunción de Ground Truthing. Identificar e informar a los investigadores de asunciones en los datos de ground truth es de vital importancia para asegurar que todos los algoritmos sean evaluados limpiamente. Por ejemplo, en la anotación de datos de los acontecimientos que ocurren en la escena sobre reconocimiento de un equipaje desatendido, uno tendría que declarar la distancia mínima y el tiempo en que el equipaje dejado es definido como desatendido en los datos de ground truth. Información respecto a los parámetros de una tarea particular de vigilancia, usados recogiendo los datos de ground truth, están disponibles desde el sitio web de Métricas PETS.

Herramientas de Anotación de Ground Truthing (GTAT). Las herramientas de GTATs son requeridas para introducir los datos de ground truth por parte de usuarios al ordenador. El primer conjunto de métricas implementadas por Métricas PETS fueron para la segmentación de movimiento. Por lo tanto, se seleccionó una herramienta para obtener el ground truth de segmentación de movimiento. Dos GTATs populares, que son disponibles gratuitamente, son Viper y la herramienta de proyecto CAVIAR. Aunque estas son herramientas sofisticadas, para Métricas PETS la GTAT de la Universidad de Reading para el proyecto de AVITRACK fue adaptada. Esta herramienta es una herramienta de anotación de descripción con predicción del movimiento lineal del objeto. Esta herramienta saca la anotación de ground truth en XML. La herramienta fue adaptada para permitir la introducción de datos de segmentación para el movimiento. Esta herramienta es mostrada en la siguiente imagen.

Ground truthing con la GTAT de la Universidad de Reading fue realizado en cuatro pasos:

1. Primero asegure que a cada objeto móvil físico único es dado una ID única y una clasificación (por ejemplo, persona, coche, etc.) que lo referencia en cada frame.

2. Siguiente, para cada objeto móvil físico único en cada frame, asegúrese que está encapsulado por una caja suficientemente grande. La predicción de movimiento lineal automática ayudó a la adición rápida de cajas para seleccionar frames precedentes o sucesivos.

3. Siguiente, segmenta cada objeto en su caja desde el background mediante identificación de píxeles divisorios de foreground sobre el objeto y usando la característica de segmentación de foreground para llenar los píxeles dentro del límite.

4. Siguiente, usa la característica de caja de apretado automático para asegurar la caja solamente extendiendo a los valores de dimensión x-y máximos y mínimos de los datos de foreground del objeto.

Métricas.-

Actualmente Métricas PETS tiene cuatro métricas implementadas para llevar a cabo la evaluación de los algoritmos de Segmentación de Movimiento. Estas métricas son las siguientes:

- Métrica de Ratio Negativo (NR)
- Métrica de Penalización de Clasificación Errónea (MP)
- Métrica de Ratio de Clasificaciones Erróneas (RM)
- Métrica de Medida de Calidad Ponderada (WQM)

Para todas las métricas de Segmentación de Movimiento, cuanto más bajo es el marcador, mejor es el algoritmo de segmentación que casa con la segmentación de foreground del ground truth. Todas las métricas son la suma de dos partes: un marcador positivo falso y un marcador negativo falso. Un marcador positivo falso bajo describe una buena identificación del límite del objeto. Un marcador negativo falso bajo describe una buena identificación del foreground interno al objeto.

La primera métrica, **Métrica de Ratio Negativo (NR),** mostrada en las ecuaciones de la 1 a la 3, evalúa un ratio negativo falso (NR_{fn}) y un ratio positivo falso (NR_{fp}). Esta métrica está basada en unos desajustes de píxel entre el ground truth y observaciones en un frame.

$$NR = NR_{fn} + NR_{fp} \quad (1)$$

donde

$$NR_{fn} = N_{fn} / N_{tp} + N_{fn} \quad (2)$$

$$NR_{fp} = N_{fp} / N_{fp} + N_{tn} \quad (3)$$

donde N_{fn} y N_{fp} denotan el número de píxeles negativos falsos y positivos falsos respectivamente. N_{tn} y N_{tp} son el número de negativos reales y positivos reales. Debería ser notado que la Métrica de Ratio Negativo puede ser usada únicamente para dar una indicación general de segmentación de objeto específica.

La segunda métrica, **Métrica de Penalización de Clasificación Errónea (MP)**, mostrada en las ecuaciones de la 4 a la 6, evalúa una segmentación de objeto del algoritmo frente al ground truth sobre una base de objeto-por-objeto. Los píxeles

clasificados de manera errónea son penalizados por sus distancias desde el borde de objeto referenciado de ground truth.

$$MP = MP_{fn} + MP_{fp} \quad (4)$$

donde

$$MP_{fn} = (\sum_{j=1}^{Nfn} d^{j}_{fn})/ D \quad (5)$$

$$MP_{fp} = (\sum_{j=1}^{Nfp} d^{j}_{fp})/ D \quad (6)$$

donde d^{j}_{fn} y d^{j}_{fp} son las distancias del píxel j^{th} negativo falso y el píxel k^{th} positivo falso desde el contorno de la segmentación de referencia. El factor normalizado D es la suma sobre todas las distancias píxel-a-contorno de objetos en un frame. Esta métrica describe cómo de bien un algoritmo puede extraer un objeto físico específico mediante la penalización de píxeles erróneamente clasificados basándose en la distancia de un límite de objeto. Si un algoritmo tiene un marcador *MP* bajo es bueno en la identificación de un limite de objeto y en la segmentación de objeto físico de la escena.

La tercera métrica, **Métrica de Ratio de Clasificaciones Erróneas (RM)**, mostrada en las ecuaciones de la 7 a la 9, evalúa una distancia promedio de píxel segmentado erróneamente del algoritmo a un borde de objeto en unidades de píxeles.

$$RM = RM_{fn} + RM_{fp} \quad (7)$$

donde

$$RM_{fn} = 1/N_{fn}\sum_{j=1}^{Nfn} d^{j}_{fn}/D_{diag} \quad (8)$$

$$RM_{fp} = 1/N_{fp}\sum_{k=1}^{Nfp} d^{k}_{fp}/D_{diag} \quad (9)$$

N_{fn} y N_{fp} denotan el número de píxeles negativos falsos y positivos falsos respectivamente. D_{diag} es la distancia diagonal del frame. Esta métrica es similar a la métrica MP pero, en este caso, usa el número de píxeles de N_{fn} o N_{fp} como el factor normalizador para RM_{fn} y RM_{fp}, respectivamente, opuesto a D usado en ecuaciones 5 y 6. Esta métrica evaluará el grado medio de error cuando se producen errores, más que la cantidad media de error que se produce.

La cuarta métrica, **Métrica de Medida de Calidad Ponderada (WQM)**, mostrada en las ecuaciones de la 10 a la 14, cuantifica la discrepancia espacial entre la segmentación de movimiento estimada y la segmentación de movimiento de objeto referenciada en el ground

truth. Esta es medida como la suma de los efectos ponderados de los píxeles positivos falsos y negativos falsos.

$$WQM = WQM_{fn} + WQM_{fp} \quad (10)$$

donde

$$WQM_{fn} = 1/N \sum_{j=1}^{Nfn} w_{fn}(d^j_{fn}) d^j_{fn} \quad (11)$$

$$WQM_{fp} = 1/N \sum_{k=1}^{Nfp} w_{fp}(d^k_{fp}) d^k_{fp} \quad (12)$$

donde N es el área del objeto de referencia en píxeles. Siguiendo el argumento en el trabajo de Aguilera *et al.* donde se dice que la importancia visual de positivos falsos y negativos falsos no es la misma, y así debería ser tratado diferentemente, la funciones de ponderación w_{fp} y w_{fn} son usadas, donde:

$$w_{fp}(d_{fp}) = B_1 + B_2/(d_{fp} + B_3) \quad (13)$$

$$w_{fn}(d_{fn}) = C\, d_{fn} \quad (14)$$

Métricas PETS usa los mismos valores constantes para B_1, B_2, B_3 y C, a saber: B_1 = 19, B_2 = -178.125, B_3 = 9.375 y C =2. La ponderación con estos valores de parámetros favorece los algoritmos que proporcionan estimaciones de foreground más grandes sobre los más conservadores.

5. VALIDACIÓN / EXPERIMENTACIÓN

En esta sección explicamos detalladamente cómo realizamos la evaluación de los resultados que obtenemos de nuestra experimentación.

En primer lugar, se detallará el Dasatet sobre el que se realiza la evaluación. A continuación, explicaremos todo el proceso que existe desde que ejecutamos nuestra aplicación sobre un vídeo determinado haciendo uso de uno de los algoritmos, hasta que se obtienen los resultados del mismo aplicando la métrica deseada. Y, por último, nos centraremos en la evaluación de resultados obtenidos.

5.1. Descripción del Dataset a evaluar

Como ya se dijo en la sección anterior, para abordar el proceso de evaluación de los algoritmos tomamos como referencia la web de PETS Metrics.

En dicha web, se proporciona el vídeo sobre el que se realizará la evaluación. Se trata del vídeo Camara1 del Dataset 1 de PETS2001. Hay que decir aquí, que este vídeo esta formado realmente por 2688 frames, pero en nuestra experimentación, la aplicación de la que disponemos procesa únicamente los primeros 1604 frames del vídeo. Esto es debido al tamaño del mismo, se ha podido comprobar como nuestra aplicación procesa vídeos de un máximo de 2 GB. Una posible solución para poder procesar la totalidad del vídeo es reducir la resolución de los frames que lo componen, pero, por el contrario, la detección empeora. Por eso, se ha decidido que nuestra evaluación se realice sobre los primeros 1604 frames del Dataset en cuestión.

Como ya se comentó en la sección 3, donde se analizaba la experimentación, nuestro vídeo está grabado en un campus universitario. En él se puede diferenciar claramente un edificio, una zona ajardinada con árboles, y otra zona asfaltada para el paso de vehículos y personas, donde también existe una zona habilitada para estacionar dichos vehículos. La siguiente imagen muestra dicho escenario:

La imagen mostrada, además de servir para mostrar el escenario, es el primer frame del vídeo, y nos servirá como punto de partida para poder contar todo lo que sucede a lo largo de los 1604 frames que lo componen.

Durante los primeros frames únicamente se aprecia el movimiento de las ramas del árbol que se encuentra situado más a la izquierda de la imagen. A los pocos instantes, simultaneándose con el movimiento de las ramas, aparece una persona en la escena por la parte izquierda en la zona asfaltada.

Dicha persona seguirá caminando en línea recta, sin pararse en ningún momento. Cuando llegue a la altura del último coche de la fila, aparecerá por la zona derecha un vehículo.

La persona seguirá andando hasta desaparecer de la escena por la parte derecha, mientras que el vehículo se dirigirá a estacionarse en el hueco de la esquina (donde aparece la persona en la imagen anterior). En este transcurso de tiempo, la persona y el vehículo llegan a cruzarse. Justo en el momento en el que el vehículo se para por completo, aparece por la izquierda una furgoneta.

Cuando la furgoneta llega a la altura del coche estacionado aparecen en la escena un grupo de personas por la zona derecha asfaltada, y por la zona izquierda ajardinada otro chico.

La furgoneta circula hasta llegar al final de la zona asfaltada, donde se para. El chico que apareció por la izquierda camina hasta la zona donde se para la furgoneta, mientras que el grupo de chicos de dirige hacia la zona ajardinada situada detrás de los vehículos estacionados. Además, la persona que estacionó el primer vehículo, sale de él, formando así parte de la escena.

El chico que aparece llegando a la furgoneta continúa su camino hasta desaparecer de la escena por la parte derecha. Hacia la furgoneta, precisamente, se dirige la persona que salió del vehículo, mientras que el grupo de jóvenes prosigue por la zona ajardinada camino del edificio.

La persona que salió del vehículo sigue caminando hasta desaparecer de la escena, a la vez que el grupo de personas continúa caminando hacia el edificio. A la altura de la furgoneta, todavía parada, aparecen un par de personas.

La pareja que acaba de aparecer en la escena camina junta por el asfalto, y el grupo de chicos continúan su camino. Cuando la pareja sobrepasa el vehículo que fue estacionado durante el vídeo, la furgoneta retoma el movimiento marcha atrás. En ese instante finaliza la evaluación del vídeo.

Esta última imagen se corresponde con el último frame (el frame 1604). Todo este conjunto de frames, del 1 al 1604, son objeto de nuestra evaluación.

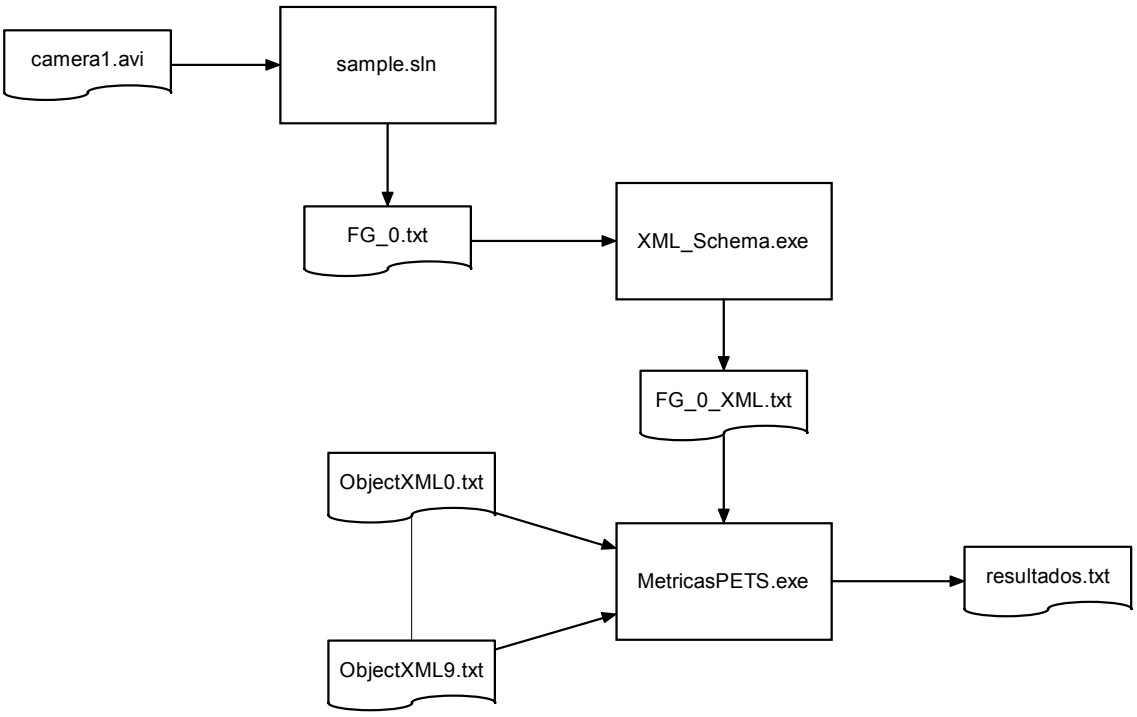
Durante todos los frames que componen el vídeo se pueden apreciar distintos aspectos de gran interés para el tema que tratamos. A modo de síntesis se pueden mencionar los siguientes:

- Foreground dinámico: Se observan objetos de foreground que salen a escena y se mueven a lo largo del vídeo. Es el caso de las personas.
- Foreground estático: También se da el caso de objetos que aparecen en la escena en movimiento, y transcurrido un tiempo se paran. Es lo que ocurre con los dos vehículos.
- Background dinámico: Se observan partes del background que se mueven debido a la presencia de viento, así como zonas que varían ligeramente su color debido a la variación de iluminación. Es el caso de las ramas del árbol y la fachada del edificio, respectivamente.
- Background estático: Evidentemente, la mayoría de los píxeles que forman el entorno, y no sufren ningún tipo de variación.

5.2. Obtención de los resultados

Una vez que ya disponemos de un vídeo sobre el que vamos a evaluar nuestros algoritmos, el siguiente paso será obtener los resultados de aplicar los algoritmos sobre dicho vídeo, para posteriormente aplicar a estos resultados las métricas vistas anteriormente.

A modo de resumen, mostramos a continuación un esquema que muestra todo este proceso de obtención de los resultados:



Este esquema representa todo el proceso de obtención de resultados, donde se puede observar como partiendo de un fichero de vídeo llegamos a conseguir un fichero de texto con los resultados que queremos evaluar. A continuación vamos a ir detallando este proceso paso a paso, tomando como criterio de ordenación las aplicaciones que se han utilizado:

sample.sln

Esta aplicación toma como entrada el vídeo sobre el que se quiere realizar la evaluación.

Esta aplicación, de la que ya se habló en secciones anteriores, no es otra que nuestra aplicación en Microsoft Visual Studio sobre la que realizamos la experimentación de nuestros dos algoritmos. Es en ella donde especificamos que algoritmo (FGD o MOG) queremos utilizar, así como los valores de los parámetros propios del algoritmo elegido. También indicaremos el nombre del fichero de salida.

Como salida de nuestra aplicación, obtendremos un fichero de texto donde se da información de los objetos que han sido detectados durante el transcurso del vídeo. Concretamente, el fichero sigue un formato YAML, en el que se indica, en primer lugar, el número de frame en el que comienza la detección de cada objeto. Tras esta información, y para cada uno de los objetos detectados, se indica: la posición en el eje X, la posición en el eje Y, el ancho y el alto.

XML_Schema.exe

Esta aplicación toma como entrada el fichero de texto resultante de la aplicación anterior.

XML_Schema.exe se encarga de convertir un fichero de texto con el formato YAML, descrito anteriormente, a un fichero de texto con el formato XML que recomienda PETS, y que ya vimos en la sección anterior. Cuando vayamos a ejecutar esta aplicación por línea de comando, tendremos que especificar el nombre que tiene el fichero de entrada, el nombre que queremos dar al fichero de salida, así como la resolución del vídeo, es decir, el ancho y el alto en píxeles. Estos dos últimos datos son necesarios porque en el fichero de entrada la información que se da acerca de los objetos no tiene asociada una unidad de medida, si no que se da en proporción, por lo que el ancho y alto del vídeo será necesario para poder convertir esa información en unidades de píxel.

Por tanto, como salida de esta aplicación, obtendremos un fichero de texto con el formato XML aportado por PETS, y que contendrá la misma información que el fichero de entrada, pero con una estructura mucho más clara. El fichero estará ordenado por número de frame, y por cada frame se irá mostrando la información de interés de los objetos que hayan sido detectados.

MetricasPETS.exe

Esta aplicación toma como entradas diferentes archivos de texto. Por un lado, el fichero de texto resultante de la aplicación anterior, donde se aporta información de los resultados de nuestra experimentación. Y por otro lado, una serie de archivos de texto, concretamente diez, que se corresponden con el número de objetos que aparecen en la escena durante el

transcurso del vídeo. De estos últimos ficheros hay que reseñar que siguen un formato XML diferente al propuesto por PETS, y que cada uno incluye información de cada uno de los objetos, concretamente aporta la información considerada ideal o perfecta de la detección de cada uno de ellos.

La aplicación MetricasPETS.exe se encargará de aplicar las métricas utilizadas por PETS sobre nuestros resultados. Para ello, tendrá que comparar el fichero de los resultados obtenidos de nuestra experimentación con los resultados ideales que se reflejan en los otros diez ficheros, y que constituyen el ground truth del vídeo. Para evaluar los resultados se ofrecen las cuatro métricas que se explicaron en la anterior sección, y que son las que considera PETS. Cuando vayamos a ejecutar esta aplicación por línea de comando, tendremos que especificar el nombre que tiene el fichero de entrada que queremos evaluar, el nombre de la métrica que queramos aplicar (las métricas NR, MP, RM y WQM), así como la resolución del vídeo, es decir, el ancho y el alto en píxeles. En este caso, estos dos últimos datos son necesarios porque alguna de las métricas utiliza esos valores para obtener sus resultados. Por último, habrá que incluir también en la línea de comandos, cuando vayamos a ejecutar la aplicación, el nombre que queremos dar al fichero que contenga los resultados de aplicar la métrica deseada.

Como salida de esta aplicación, obtendremos un fichero de texto con los resultados de aplicar las métricas sobre nuestros resultados. En concreto, el fichero tendrá tantas líneas como frames tenga el vídeo, y en cada una de ellas se mostrará el valor que se obtiene de aplicar la métrica deseada en ese frame.

5.3. Evaluación de los resultados

Tras realizar todo el proceso anterior de obtención de los resultados, nos encontramos en una situación en la que disponemos de un fichero de texto con los resultados de aplicar una métrica concreta sobre los resultados de nuestros algoritmos. Pero claro, esos resultados vienen dados para cada uno de los frames que conforman el vídeo, es decir, hay tantos valores como frames hay en el vídeo.

El siguiente paso será, por lo tanto, interpretar esos resultados obtenidos. Se tratará de obtener un valor cuantitativo para cada una de las métricas aplicadas que nos permitan comparar unos resultados con otros, para llegar a la conclusión de en qué casos de obtiene una mejor aproximación al resultado considerado ideal o perfecto o, lo q es lo mismo, al ground truth.

A continuación, iremos evaluando los resultados para cada uno de los dos algoritmos. De esta forma, para cada uno de ellos, en función de toda la experimentación realizada, mostraremos los resultados que se consideraron mejores y los compararemos entre sí para ver cuando se presenta un mejor comportamiento. Finalmente, se compararán los dos algoritmos entre sí.

Algoritmo FGD

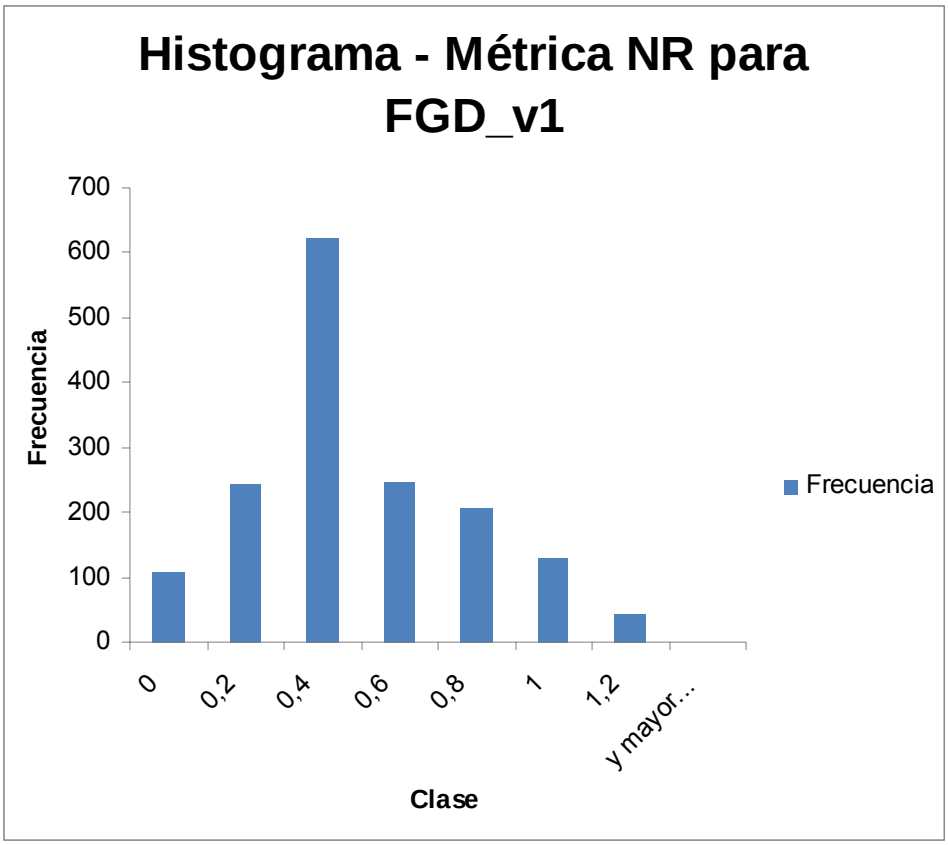
Para este algoritmo vamos a considerar, en un primer lugar, dos versiones diferentes, que tras realizar la experimentación, se considera que pueden presentar unos mejores resultados. Para cada una de ellas, mostraremos los valores dados a los parámetros propios del algoritmo, así como los resultados de aplicar cada una de las métricas a todos los frames del vídeo. De esta forma, para cada métrica, presentaremos una gráfica (histograma) donde se mostrarán los resultados de aplicar dicha métrica, al igual que se concluirá con un promedio de los valores obtenidos, que nos servirá como valor cuantitativo para ser comparado con otros resultados. Las dos versiones para este primer algoritmo se presentan a continuación:

- **FGD_v1**: Comenzaremos por evaluar los resultados obtenidos cuando asignamos a los parámetros los valores que se consideraron mejores durante toda la etapa de experimentación, y que se corresponden con los valores por defecto del algoritmo. Estos son los siguientes:

FGD
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80

A continuación, evaluamos los resultados obtenidos. Para ello, analizamos los resultados que se obtienen de aplicar cada una de las métricas:

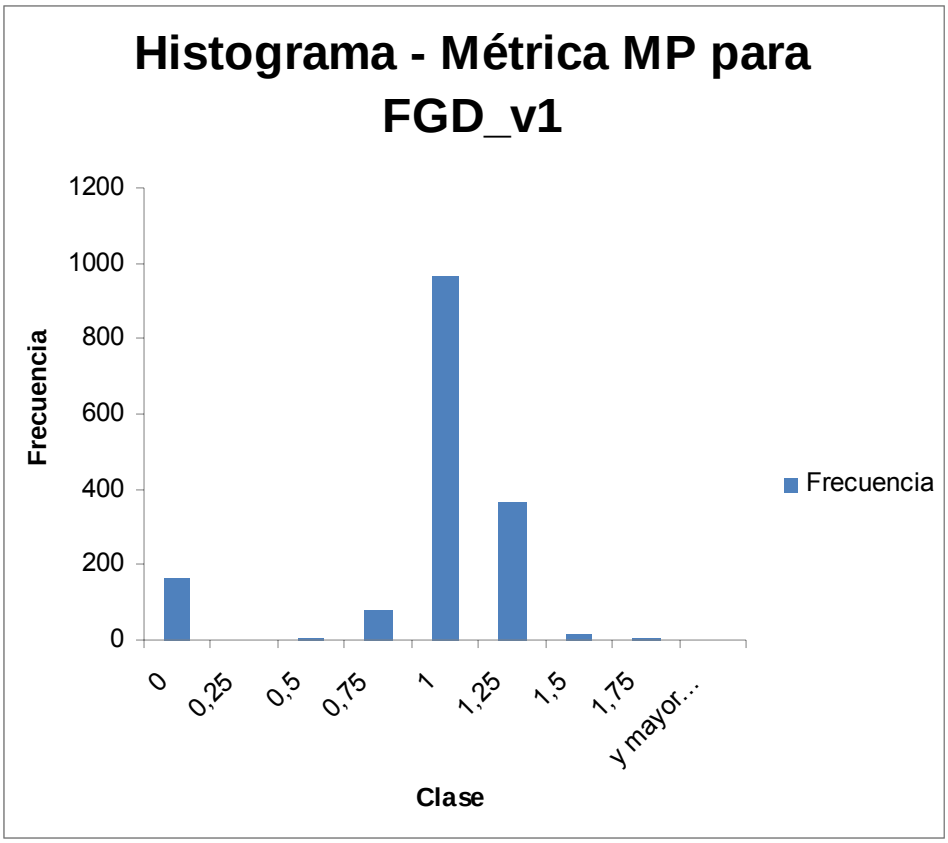
  - *Métrica NR*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	109
0,2	244
0,4	624
0,6	246
0,8	206
1	131
1,2	44
y mayor...	0

PROMEDIO = 3,95E-01

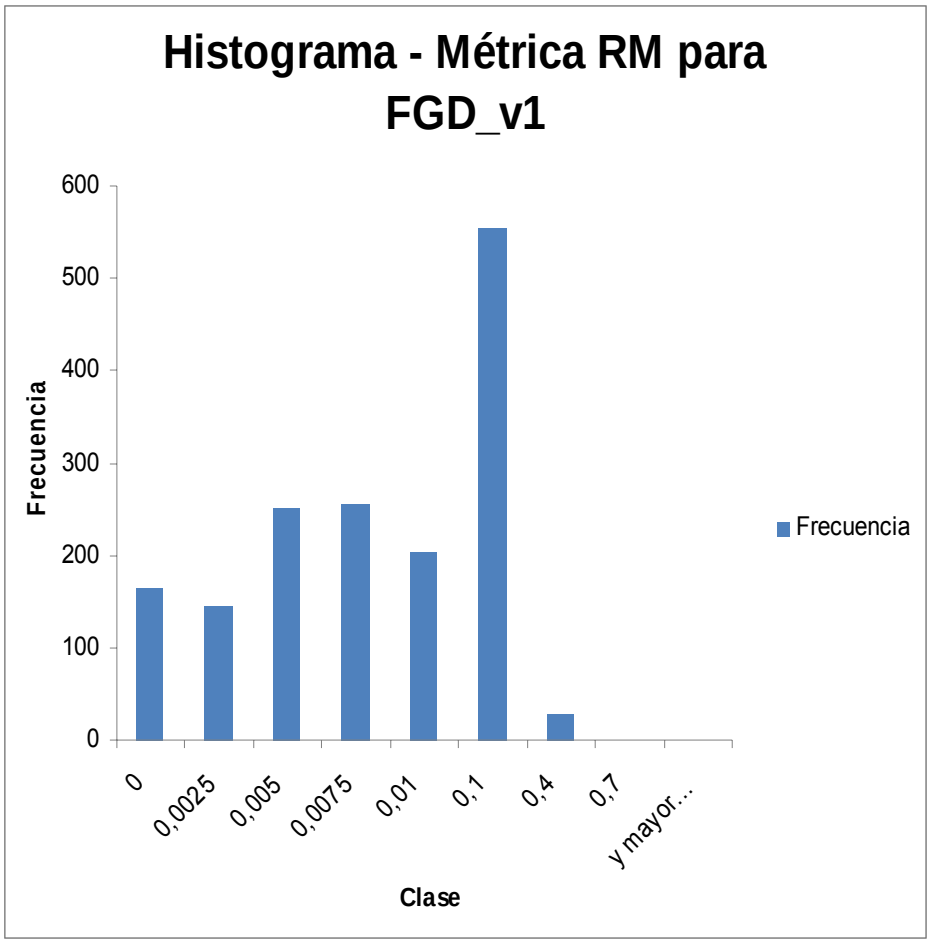
- *Métrica MP*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	164
0,25	0
0,5	7
0,75	79
1	969
1,25	364
1,5	18
1,75	3
y mayor...	0

PROMEDIO = 8,66E-01

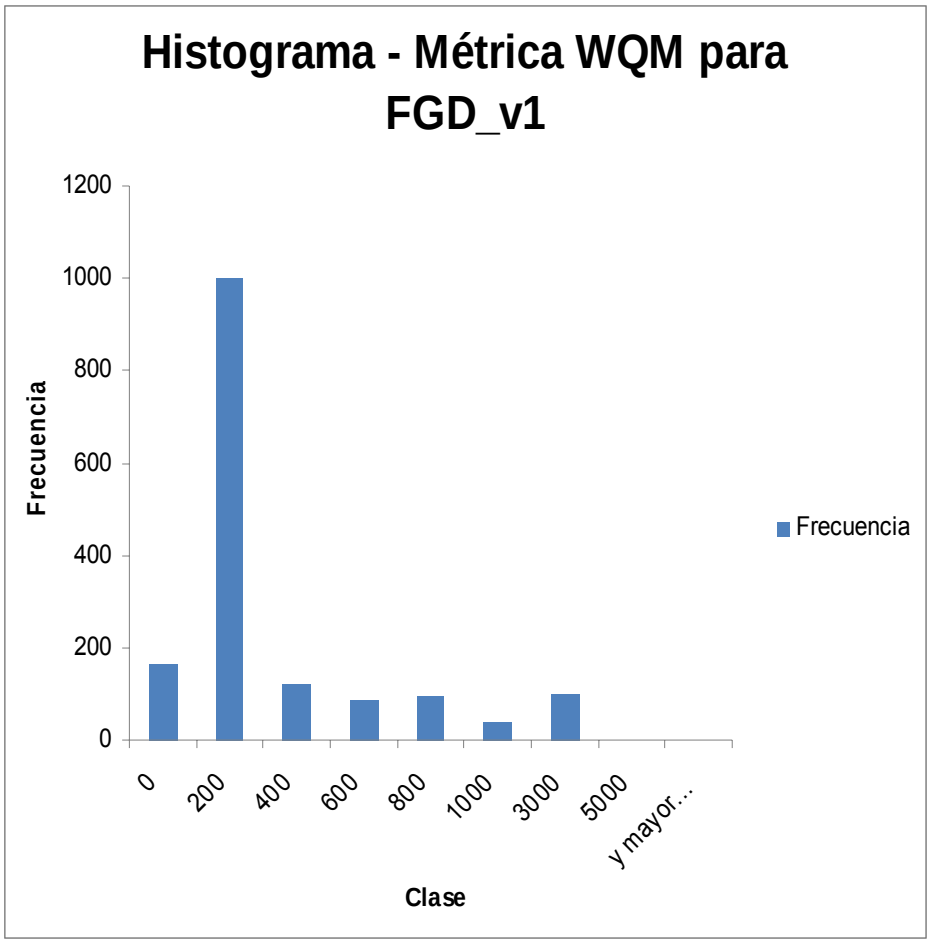
- *Métrica RM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	164
0,0025	146
0,005	252
0,0075	255
0,01	204
0,1	555
0,4	28
0,7	0
y mayor...	0

PROMEDIO = 1,51E-02

- *Métrica WQM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	164
200	999
400	121
600	87
800	97
1000	37
3000	99
5000	0
y mayor...	0

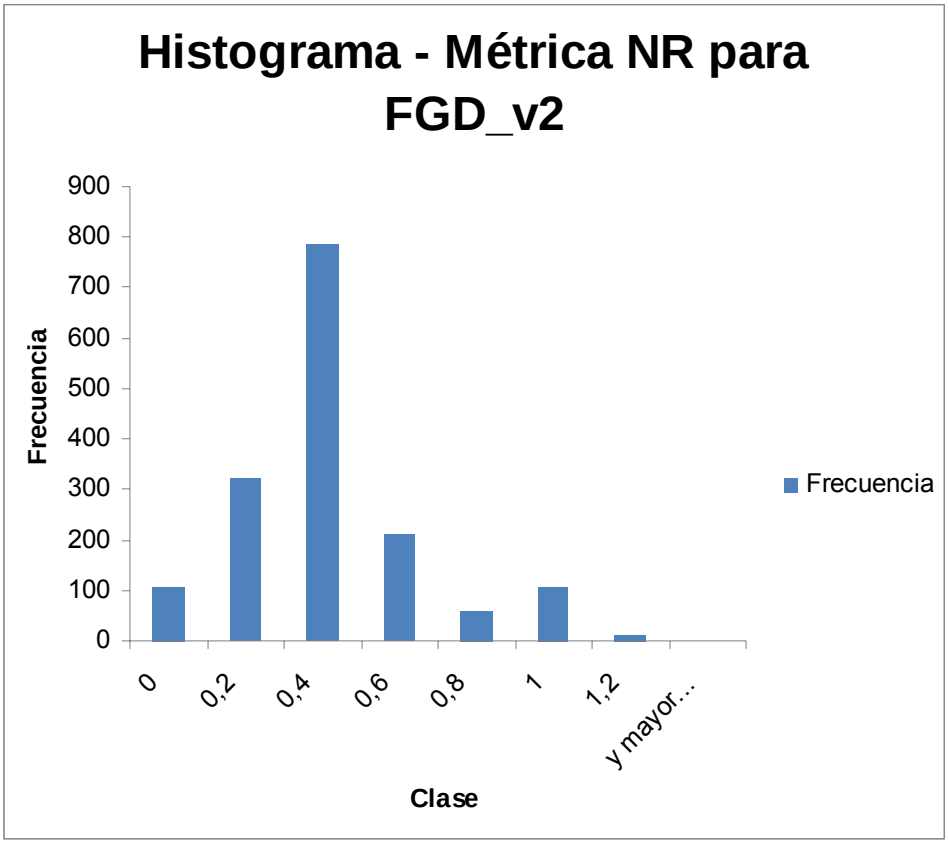
PROMEDIO = 2,41E+02

- **FGD_v2**: En esta segunda versión, variamos únicamente uno de los parámetros, concretamente el parámetro *alpha2*. Como ya se explicó en la experimentación, cuando aumentamos el valor de este parámetro, se disminuye el tiempo que se tarda en asimilar un objeto como background cuando éste se detiene y pasa a formar parte del background estático. De esta forma, daremos el valor 0.1 al parámetro *alpha2*, en lugar del valor 0.005 que tenía por defecto. Los parámetros de esta segunda versión son, por tanto los siguientes:

FGD
Lc= 64 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.1 *alpha3*= 0.1 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80

A continuación, evaluamos los resultados obtenidos. Para ello, analizamos los resultados que se obtienen de aplicar cada una de las métricas:

- *Métrica NR*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.

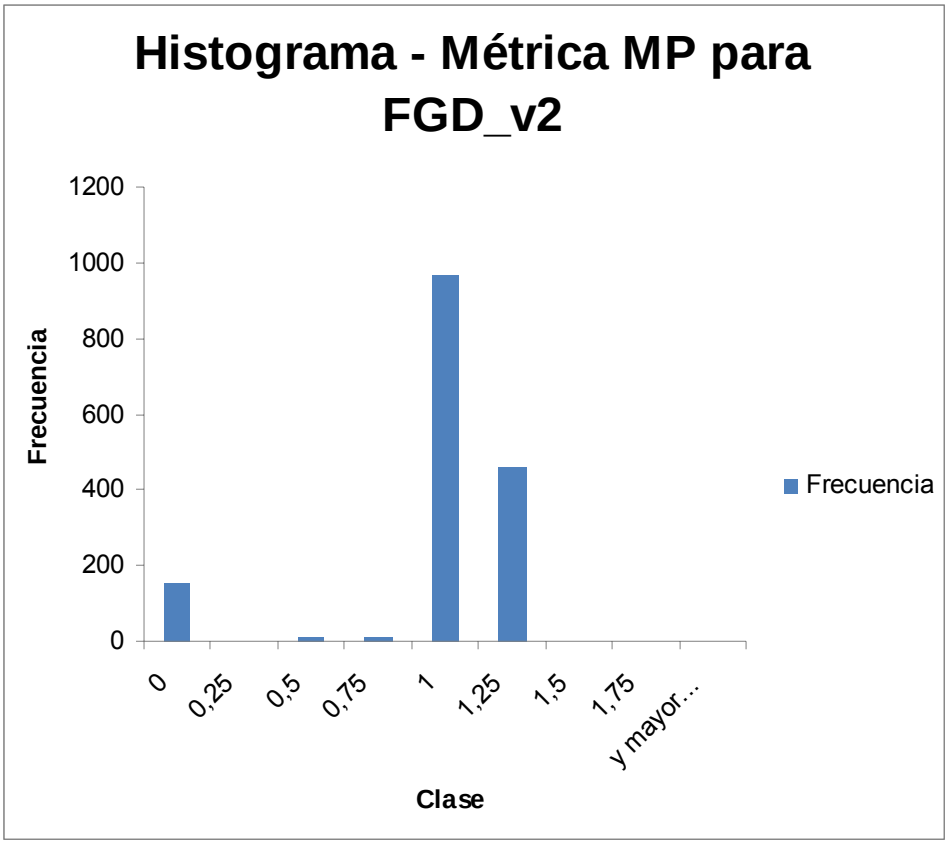


Clase	*Frecuencia*
0	109
0,2	320

0,4	786
0,6	209
0,8	61
1	109
1,2	10
y mayor...	0

PROMEDIO = 3,20E-01

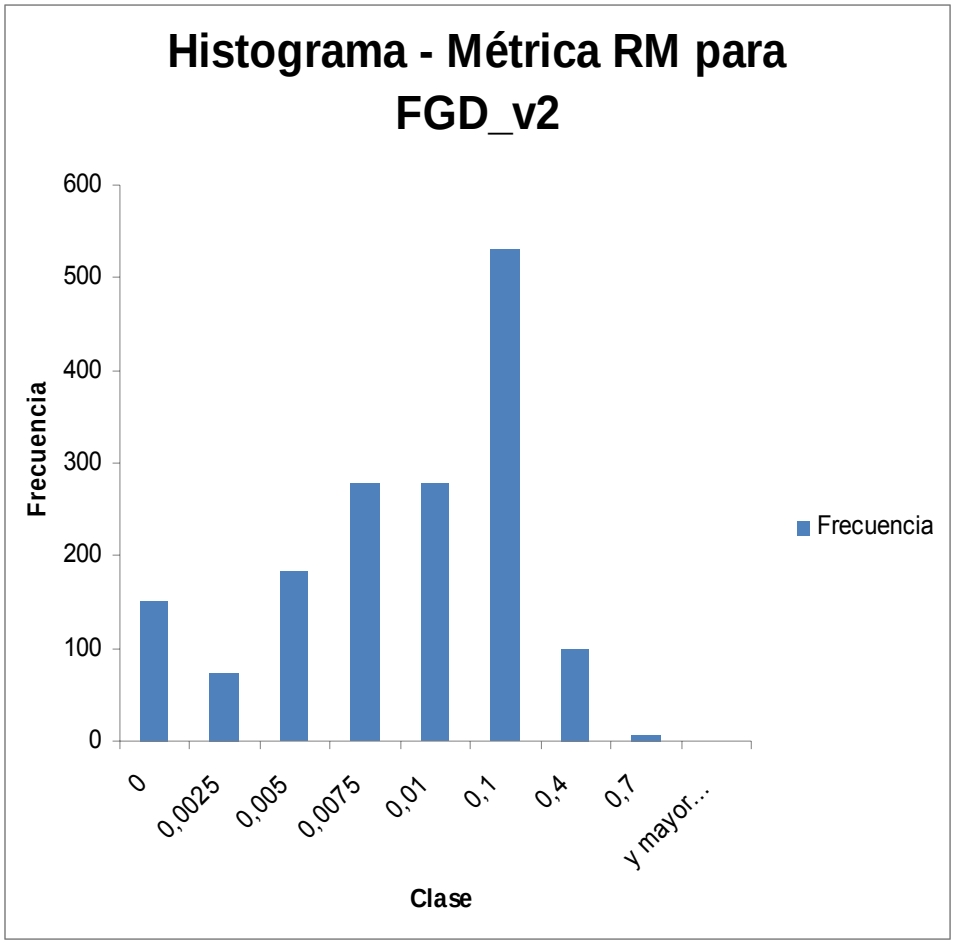
- *Métrica MP*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	151
0,25	0
0,5	8
0,75	13
1	970
1,25	462
1,5	0
1,75	0
y mayor...	0

PROMEDIO = 8,86E-01

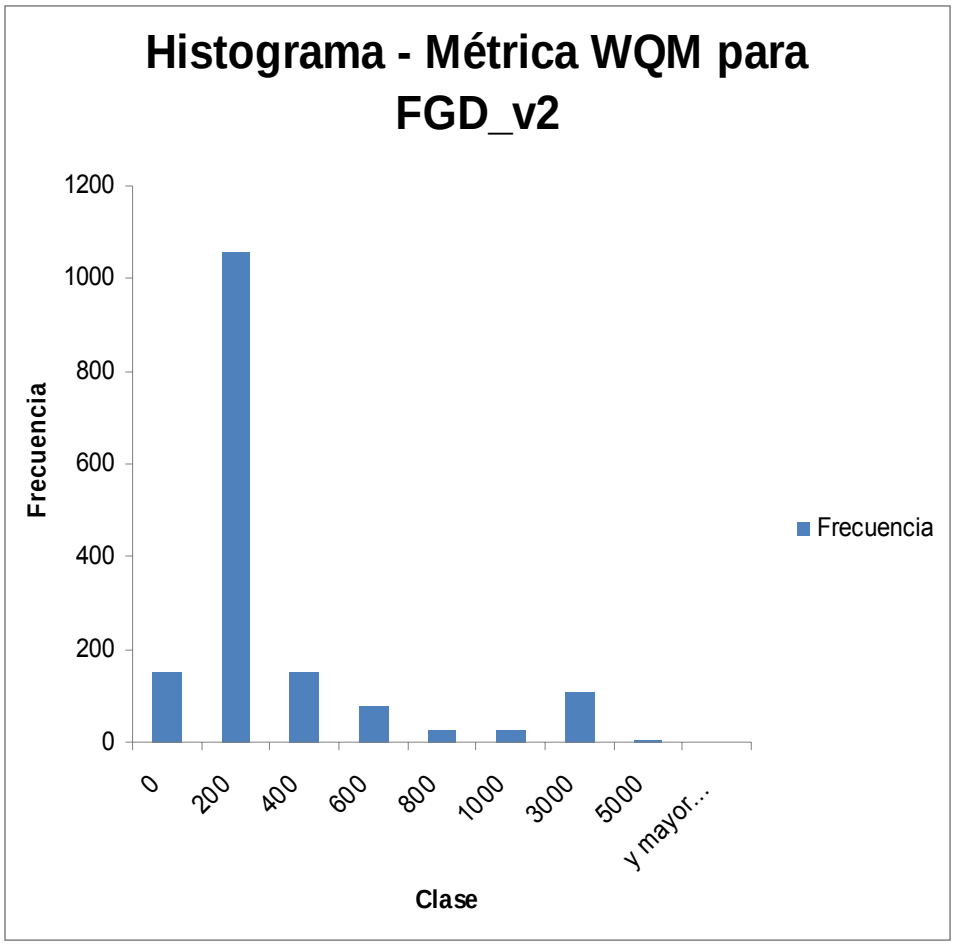
- *Métrica RM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	151
0,0025	74
0,005	183
0,0075	278
0,01	279
0,1	532
0,4	100
0,7	7
y mayor...	0

PROMEDIO = 2,83E-02

- *Métrica WQM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	151
200	1058
400	149
600	79
800	28
1000	27
3000	108
5000	4
y mayor...	0

PROMEDIO = 2,37E+02

A modo de resumen, y para analizar más fácilmente los resultados, mostramos la siguiente tabla:

	NR	MP	RM	WQM
FGD_v1	0,395	0,866	0,0151	241
FGD_v2	0,32	0,886	0,0283	237

Antes de centrarnos en comparar los resultados obtenidos, es conveniente resaltar que, a partir de ahora, daremos mayor relevancia a los resultados de la métrica NR. Se decide esto porque esta métrica da un indicador más general, que no se ve afectado por el mejor o peor comportamiento de la versión del algoritmo con el que se trabaje. Para que lo entendamos, la métrica NR en cada frame tiene en cuenta los píxeles que se detectan correctamente y los que no. Sin embargo, las otras tres métricas se dedican a comparar la detección de los objetos que detectamos con sus correspondientes del ground truth. Esto implica que, cuando no detectemos un objeto (que realmente existe) no se podrá hacer una evaluación de lo bien o mal que detectamos tal objeto, por lo que estas métricas darán, en esos casos, valor cero, lo que afecta directamente al promedio de la métrica, disminuyendo su valor. Por lo tanto, una versión que sea mala detectando objetos es posible que para alguna de esas métricas de un promedio más bajo que otra versión que detecte más objetos y su comportamiento sea mejor. Pero hay que dejar claro que esto no es una limitación, ya que bastaría con tener en cuenta los valores en los frames que son distintos de cero, y hacer la media de esos valores, y así poder comparar unas versiones con otras.

Como se puede apreciar en la tabla, se obtienen mejores resultados (valores más bajos) con **FGD_v2** con las métricas NR y WQM, obteniéndose valores algo peores con las otras dos métricas. El mejor resultado con la métrica NR confirma la idea que se tenía durante la fase de experimentación, esto es, que al disminuirse el tiempo que se tarda en asimilar como background a los objetos que detienen, el comportamiento es mejor, ya que es menor el tiempo que se sigue detectando como foreground a objetos que se han parado, o lo que es lo mismo, se reduce el número de píxeles detectados erróneamente.

De todas formas, será decisión de la persona encargada de evaluar el centrarse en una, varias o todas las métricas, en función del estudio que quiere realizar. En nuestro caso, como se ha dicho anteriormente, daremos más relevancia a los resultados de la métrica NR, pero sin olvidarnos de las otras tres.

Lo que se ha mostrado anteriormente es una tabla con los resultados de las dos versiones del algoritmo FGD que se consideraron mejores a la hora de experimentar con los diferentes vídeos. Durante esa fase de experimentación, se fueron probando diferentes valores para los distintos parámetros, y se tenía la impresión de que no se mejoraba el comportamiento, observándose en la mayoría de los casos un comportamiento muy similar. Ahora bien, esto no deja de ser una impresión. Lo ideal sería corroborar con datos cuantitativos dicha impresión. Para ello, iremos mostrando en tablas los resultados de las métricas que se obtienen cuando variamos los valores de los diferentes parámetros del algoritmo.

Comenzaremos con el parámetro *Lc* (niveles cuantizados para el componente de color). Como se dijo en su momento, este parámetro toma normalmente como valor 32, 64 o 128. El valor por defecto es 64, por lo que ahora probaremos con los otros dos valores. Mostramos los resultados en la siguiente tabla:

	NR	MP	RM	WQM
Lc = 32	0,415	0,848	0,0156	279
Lc = 128	0,383	0,87	0,016	246

El parámetro *Lcc* (niveles cuantizados para el componente de co-ocurrencia de color) toma normalmente como valor 16, 32 o 64. El valor por defecto es 32, por lo que probaremos con los otros dos valores. Mostramos los resultados en la siguiente tabla:

	NR	MP	RM	WQM
Lcc = 16	0,516	0,758	0,0136	241
Lcc = 64	0,38	0,871	0,0149	237

Los parámetros *N1c* (número de vectores de color usados para modelar la variación de color del background normal en un píxel dado) y *N2c* (número de vectores de color mantenidos en un píxel dado) están relacionados directamente, ya que el segundo tiene que ser mayor que el primero, normalmente 5/3 veces el primero. Los valores por defecto son 30 y 50, respectivamente, por lo que la idea, al igual que se hizo con los anteriores parámetros, es probar con valores menores y mayores. Aunque para el caso de valores mayores, nuestra aplicación no llega a procesar el vídeo. Por tanto, probaremos únicamente con los valores 15 y 25, que se recomendaron también en alguna ocasión. Mostramos los resultados en la siguiente tabla:

	NR	MP	RM	WQM
N1c = 15 **N2c = 25**	0,398	0,859	0,0151	239

Los parámetros *N1cc* (número de vectores de co-ocurrencia de color usados para modelar la variación de color del background normal en un píxel dado) y *N2cc* (número de vectores de co-ocurrencia de color mantenidos en un píxel dado) están relacionados directamente, ya que el segundo tiene que ser mayor que el primero, normalmente 5/3 veces el primero. Los valores por defecto son 50 y 80, respectivamente, por lo que la idea, al igual que se hizo con los anteriores parámetros, es probar con valores menores y mayores. Aunque para el caso de valores mayores, nuestra aplicación no llega a procesar el vídeo. Por tanto, probaremos únicamente con los valores 25 y 40, que se recomendaron también en alguna ocasión. Mostramos los resultados en la siguiente tabla:

	NR	MP	RM	WQM
N1cc = 25 **N2cc = 40**	0,395	0,866	0,0151	241

El parámetro *alpha1* (indica cómo de rápido se olvidan los valores de un píxel viejo de foreground) toma normalmente como valor 0,1. Consideraremos el rango (0,1) y probaremos con valores dentro de ese rango. Mostramos los resultados en la siguiente tabla:

	NR	MP	RM	WQM
alpha1 = 0,3	0,406	0,858	0,0148	242
alpha1 = 0,5	0,402	0,863	0,0151	237
alpha1 = 0,7	0,412	0,857	0,0141	231
alpha1 = 0,9	0,421	0,849	0,0136	232

El parámetro *alpha3* (suplente de alpha2, usado, por ejemplo, para una rápida convergencia inicial) toma normalmente como valor 0,1. Consideraremos el rango (0,1) y probaremos con valores dentro de ese rango. Mostramos los resultados en la siguiente tabla:

	NR	MP	RM	WQM
alpha3 = 0,3	0,397	0,864	0,0151	241
alpha3 = 0,5	0,397	0,864	0,0151	240
alpha3 = 0,7	0,395	0,866	0,0151	240
alpha3 = 0,9	0,398	0,861	0,015	240

Fijándonos en todos los resultados expuestos anteriormente y centrándonos en la métrica NR, puede observarse como con los parámetros *alpha2*, *Lc* y *Lcc* con valores 0,1, 128 y 64, respectivamente, se obtienen unos resultados mejores en comparación con la versión inicial que tenía todos los parámetros con los valores por defecto. Por tanto, el siguiente paso será evaluar una nueva versión de este algoritmo teniendo en cuenta esta mejoría en el comportamiento. De esta forma, consideramos una nueva versión del algoritmo cuyos valores para los diferentes parámetros los mostramos a continuación:

FGD
Lc= 128 *Lcc*= 64 *alpha1*= 0.1

alpha2= 0.1
alpha3= 0.1
N1c= 30
N2c= 50
N1cc= 50
N2cc= 80

Aplicando la métrica NR sobre esta nueva versión, obtenemos el siguiente resultado:

PROMEDIO = 3,01E-01

Este resultado se convierte en el mejor de todos los observados, confirmándose la idea de que mezclando versiones buenas se pueden obtener versiones mejoradas.

Algoritmo MOG

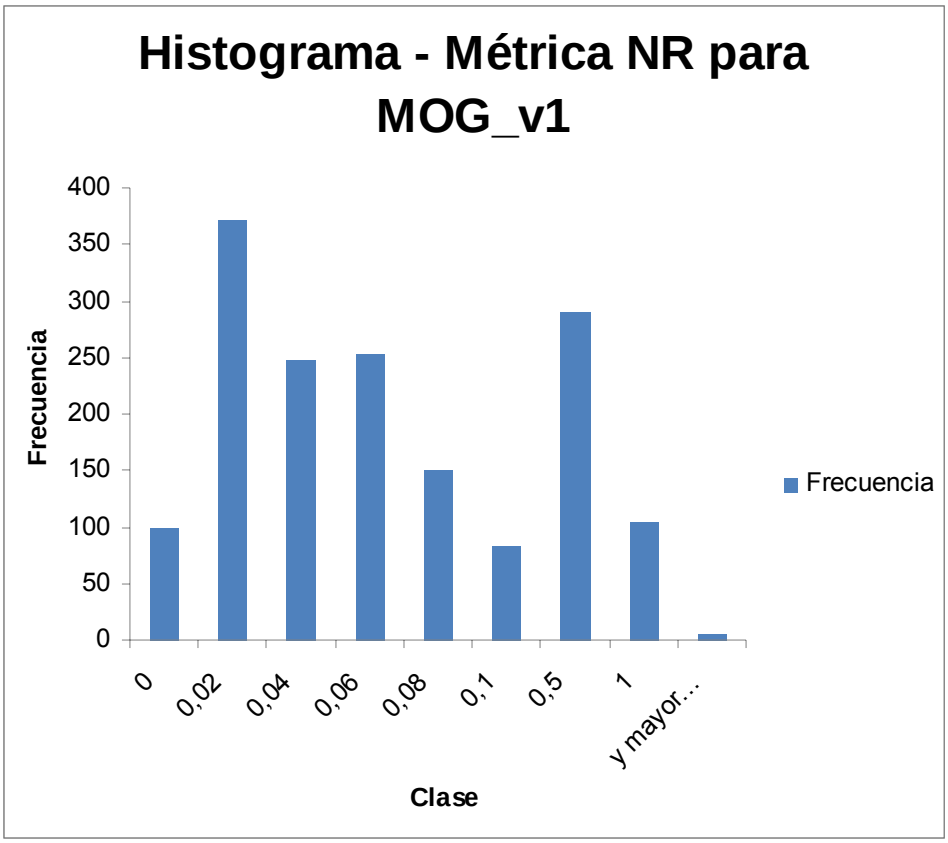
Para este algoritmo consideraremos tres versiones diferentes, que se corresponden con las recomendadas en la explicación teórica del algoritmo. Para cada una de ellas, mostraremos el valor dado al parámetro principal del algoritmo, así como los resultados de aplicar cada una de las métricas a todos los frames del vídeo. De esta forma, para cada métrica, presentaremos una gráfica (histograma) donde se mostrarán los resultados de aplicar dicha métrica, al igual que se concluirá con un promedio de los valores obtenidos, que nos servirá como valor cuantitativo para ser comparado con otros resultados. Las tres versiones para este primer algoritmo se presentan a continuación:

- **MOG_v1**: Comenzaremos por evaluar los resultados obtenidos cuando asignamos al parámetro el valor que se consideró mejor durante toda la etapa de experimentación. Éste es el siguiente:

MOG
NG= 3

A continuación, evaluamos los resultados obtenidos. Para ello, analizamos los resultados que se obtienen de aplicar cada una de las métricas:

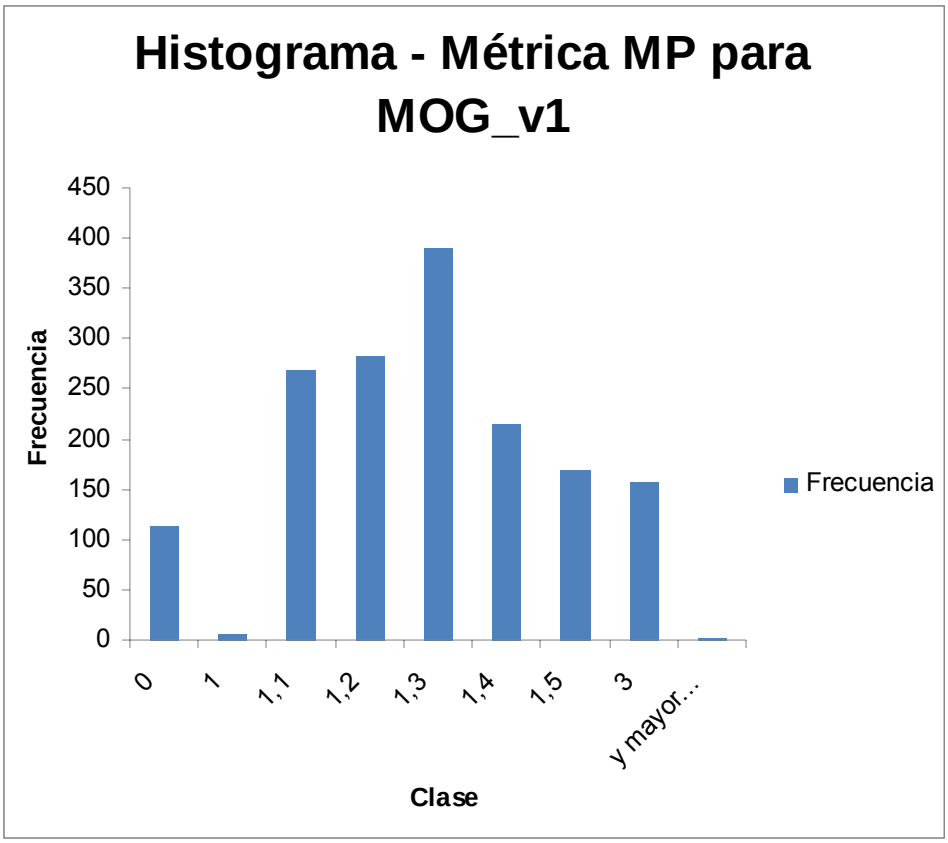
  - *Métrica NR*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	100
0,02	372
0,04	247
0,06	253
0,08	150
0,1	83
0,5	290
1	104
y mayor...	5

PROMEDIO = 1,14E-01

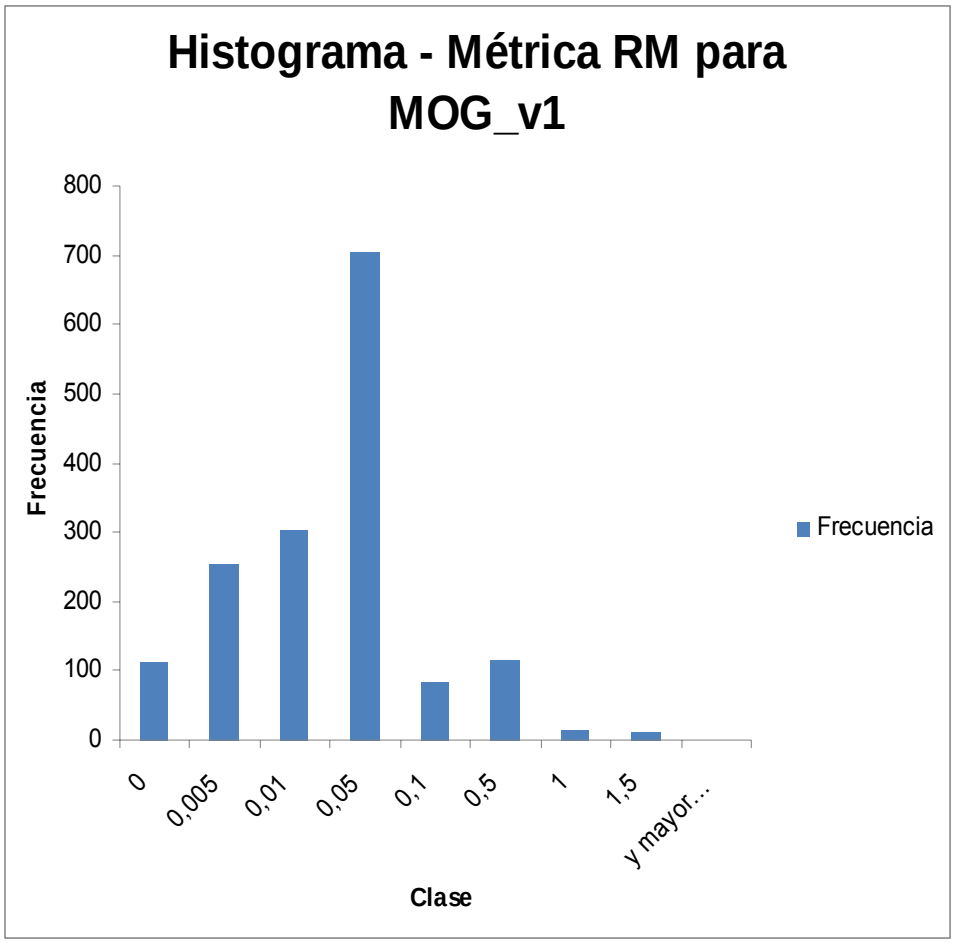
- *Métrica MP*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
1	5
1,1	269
1,2	282
1,3	390
1,4	216
1,5	169
3	158
y mayor...	1

PROMEDIO = 1,18E+00

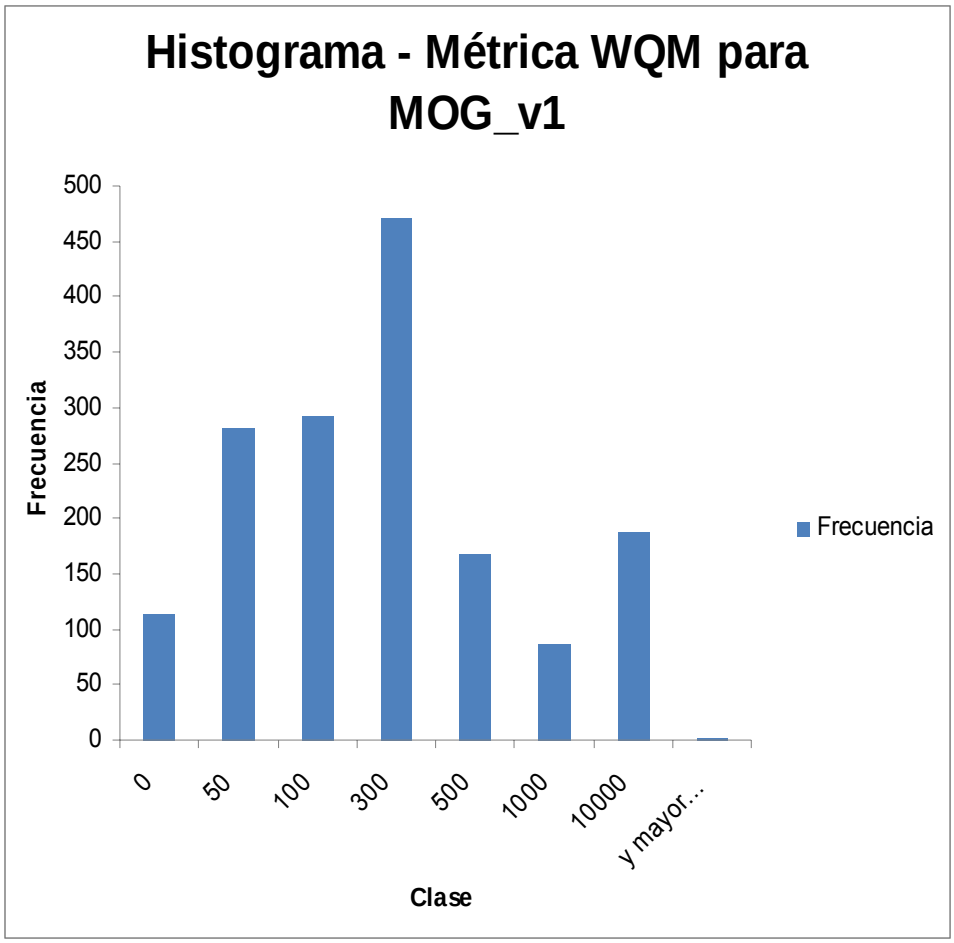
- *Métrica RM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
0,005	255
0,01	304
0,05	705
0,1	85
0,5	116
1	13
1,5	11
y mayor...	1

PROMEDIO = 4,52E-02

- *Métrica WQM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
50	282
100	293
300	472
500	167
1000	87
10000	188
y mayor...	1

PROMEDIO = 4,14E+02

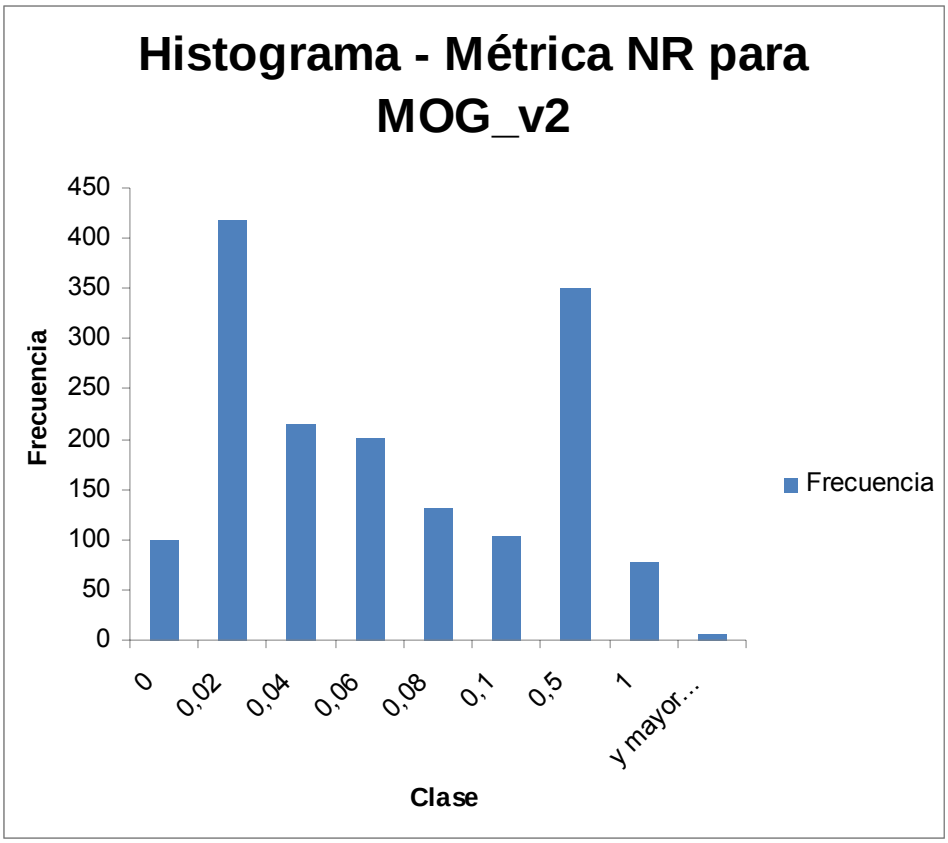
- **MOG_v2**: En esta segunda versión, probaremos con otro valor que está dentro del rango que se recomendaba en la explicación teórica del algoritmo, concretamente con el valor 4. El parámetro de esta segunda versión es, por tanto, el siguiente:

MOG

NG= 4

A continuación, evaluamos los resultados obtenidos. Para ello, analizamos los resultados que se obtienen de aplicar cada una de las métricas:

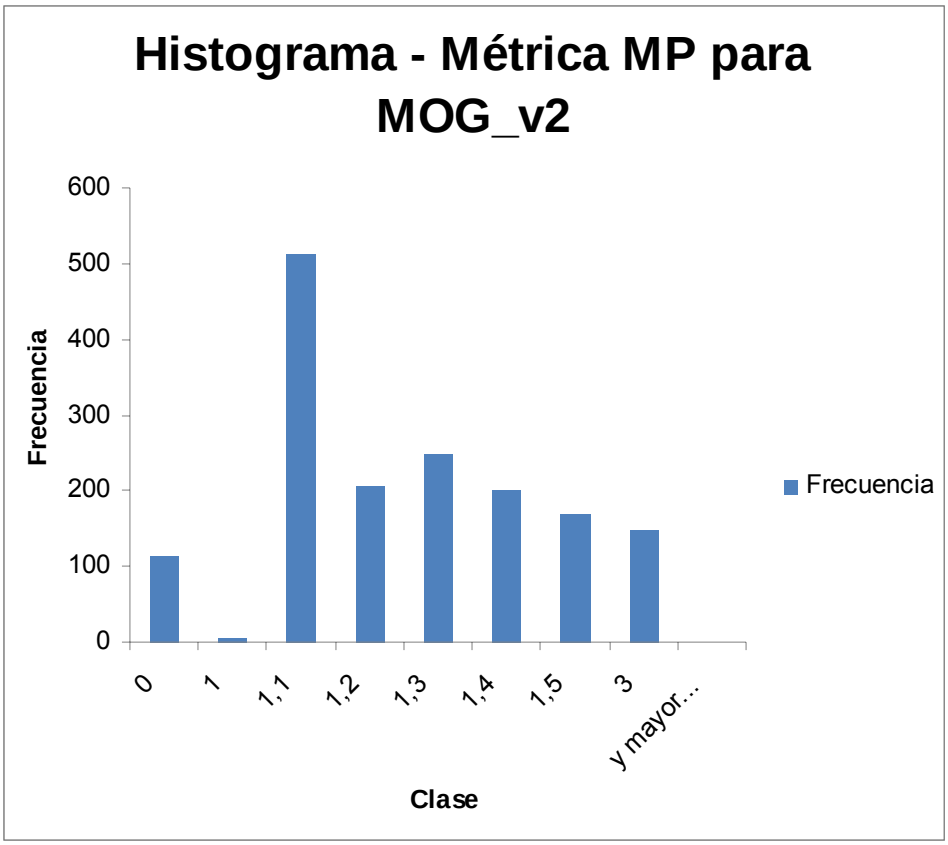
- *Métrica NR*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	100
0,02	419
0,04	216
0,06	202
0,08	131
0,1	103
0,5	350
1	78
y mayor...	5

PROMEDIO = 1,09E-01

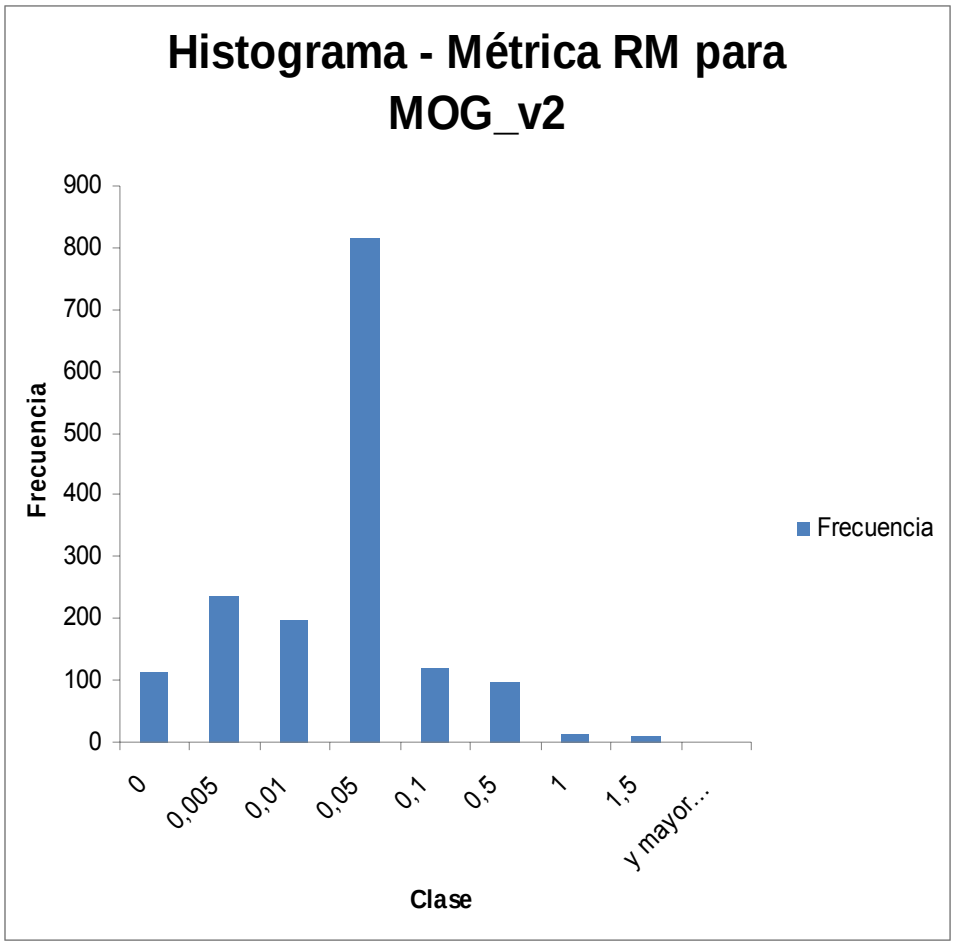
- *Métrica MP*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
1	5
1,1	514
1,2	205
1,3	248
1,4	200
1,5	170
3	147
y mayor...	1

PROMEDIO = 1,16E+00

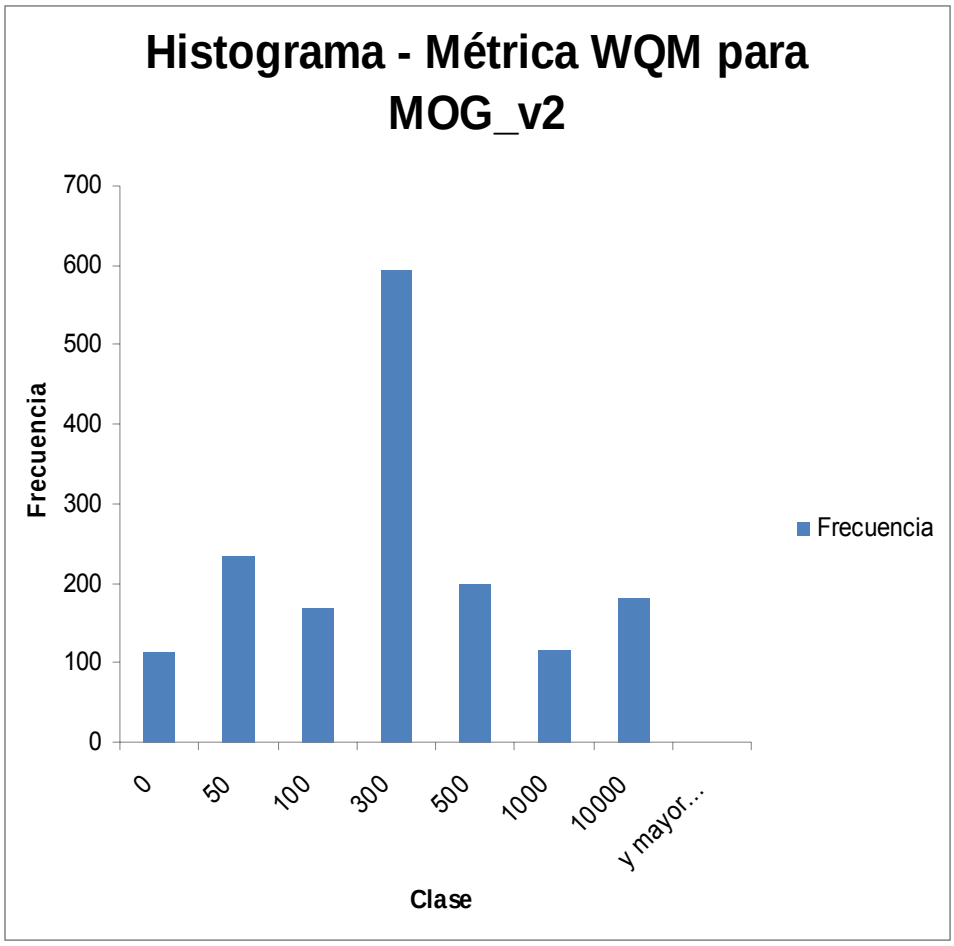
- *Métrica RM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
0,005	235
0,01	199
0,05	815
0,1	120
0,5	96
1	13
1,5	11
y mayor...	1

PROMEDIO = 4,51E-02

- *Métrica WQM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
50	233
100	168
300	593
500	199
1000	115
10000	181
y mayor...	1

PROMEDIO = 4,10E+02

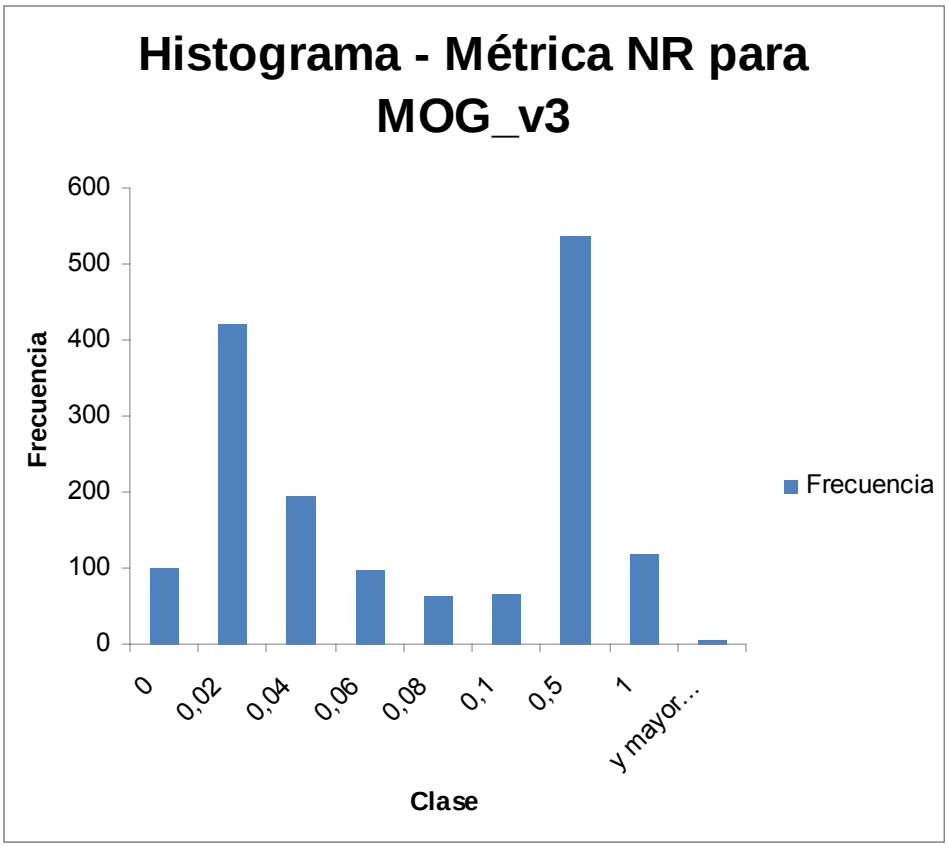
- **MOG_v3**: En esta tercera versión, probaremos con el último valor que está dentro del rango que se recomendaba en la explicación teórica del algoritmo, concretamente con el valor 5. El parámetro de esta segunda versión es, por tanto, el siguiente:

MOG

NG= 5

A continuación, evaluamos los resultados obtenidos. Para ello, analizamos los resultados que se obtienen de aplicar cada una de las métricas:

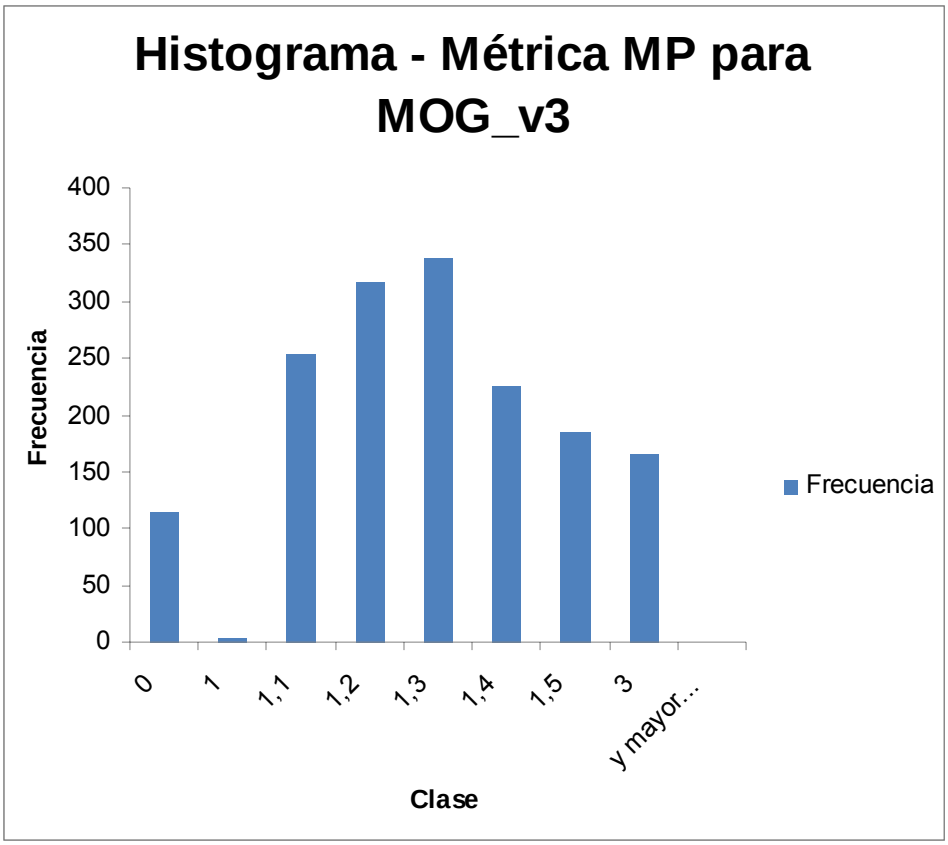
- *Métrica NR*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	100
0,02	421
0,04	196
0,06	98
0,08	64
0,1	66
0,5	536
1	118
y mayor...	5

PROMEDIO = 1,46E-01

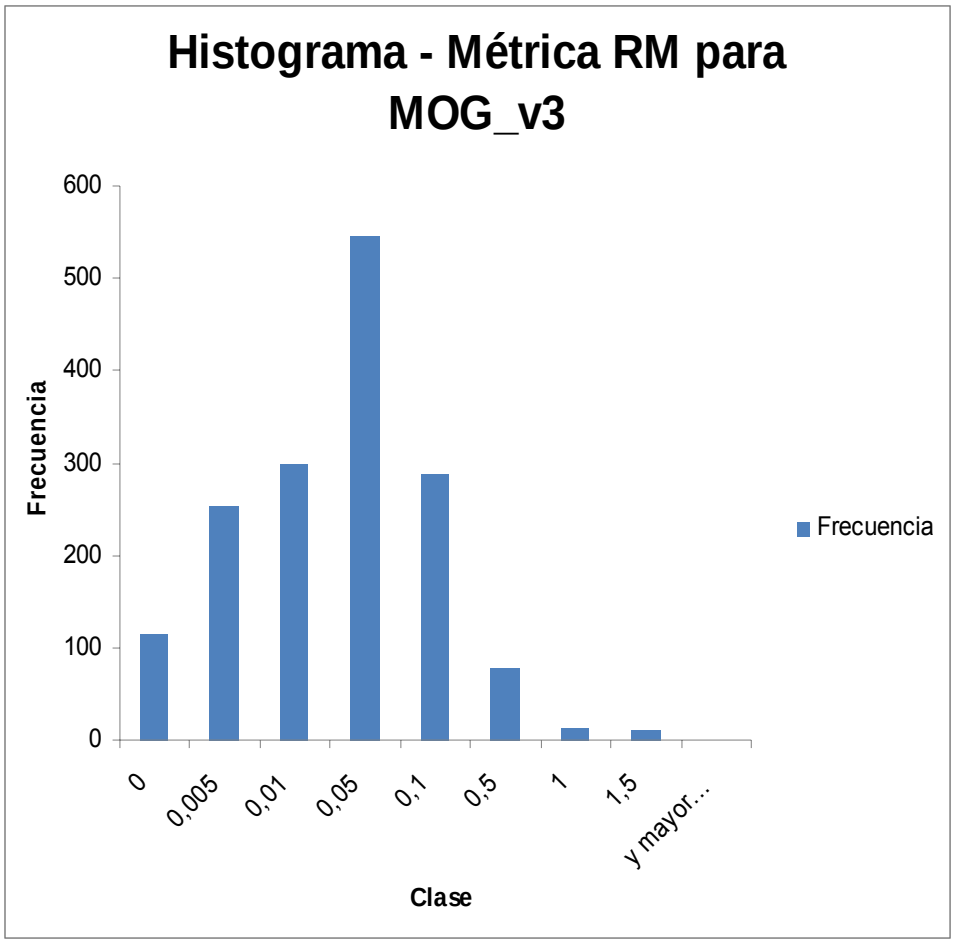
- *Métrica MP*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
1	4
1,1	254
1,2	317
1,3	339
1,4	226
1,5	185
3	165
y mayor...	0

PROMEDIO = 1,19E+00

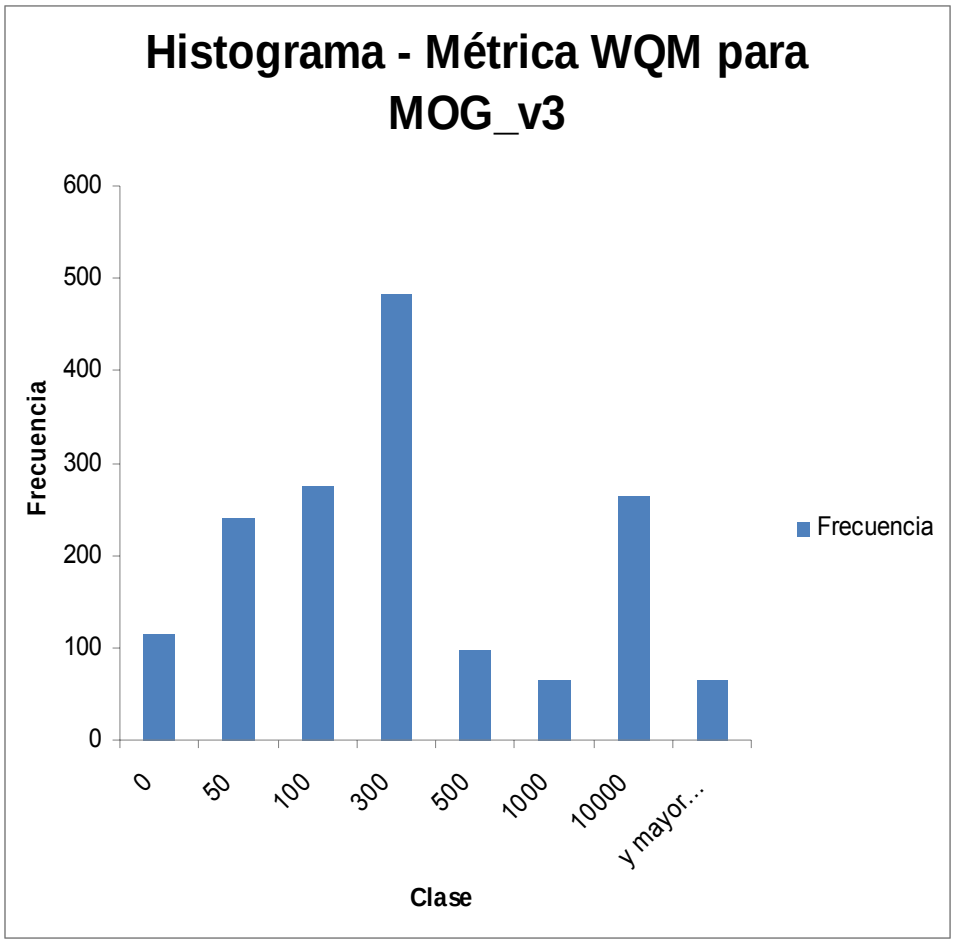
- *Métrica RM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
0,005	253
0,01	299
0,05	545
0,1	289
0,5	79
1	13
1,5	11
y mayor...	1

PROMEDIO = 4,74E-02

- *Métrica WQM*: Mostramos el histograma con los valores que se obtienen al aplicar esta métrica sobre los resultados del algoritmo.



Clase	*Frecuencia*
0	114
50	240
100	275
300	483
500	97
1000	65
10000	264
y mayor...	66

PROMEDIO = 1,32E+03

A modo de resumen, y para analizar más fácilmente los resultados, mostramos la siguiente tabla:

	NR	MP	RM	WQM
MOG_v1	0,114	1,18	0,0452	414
MOG_v2	0,109	1,16	0,0451	410
MOG_v3	0,146	1,19	0,0474	1320

Como se puede apreciar, se obtienen mejores resultados (valores más bajos) con **MOG_v2**. Esto contradice la idea que se tenía durante la fase de experimentación, cuando se hacía una evaluación visual. En este caso, con valor igual a 4 para el parámetro principal del algoritmo, se obtiene el mejor comportamiento para las cuatro métricas, si bien hay ocasiones en los que las diferencias son muy pequeñas, coincidiendo este hecho con la impresión que se tenía al experimentar con este algoritmo sobre los diferentes vídeos.

Finalmente, quedaría comparar los dos algoritmos entre sí. Para ello, partiremos de los mejores resultados que se obtienen en cada uno de ellos, y que mostramos en la siguiente tabla:

	NR	MP	RM	WQM
FGD	0,301	0,758	0,0136	231
MOG	0,109	1,16	0,0451	410

Como se puede observar en los resultados de la tabla, se prefiere el algoritmo **MOG** teniendo en cuenta cómo es la detección global a lo largo del vídeo, pero si tenemos en cuenta cómo es la detección de los objetos, se prefiere utilizar el algoritmo **FGD**.

Con estos resultados se confirma, por lo tanto, la conclusión que se obtenía de la fase de experimentación, donde se observaba un comportamiento general mejor con el algoritmo **MOG**, ya que, aunque las detecciones de los objetos por separado no fueran tan "perfectas" (por ejemplo, por considerar foreground a las sombras de los objetos), sí que era globalmente mejor teniendo en cuenta que no presentaba tantos problemas a la hora de detectar objetos que se encontraban lejanos a la cámara.

El siguiente, y último paso, será comprobar si la elección de unos determinados valores para los diferentes parámetros de los algoritmos depende del vídeo sobre el que se quiere trabajar. Para el vídeo anterior, ya hemos visto qué valores de los parámetros son los mejores. Ahora, haremos uso de otro vídeo y volveremos a experimentar sobre él con los dos algoritmos y probando diferentes valores para los parámetros propios de cada uno de ellos.

El vídeo seleccionado es uno de los analizados en la fase de experimentación. Concretamente, se trata del primer vídeo correspondiente al primero de los entornos analizados, es decir, el entorno del interior de un edificio. Durante el transcurso de dicho vídeo se pueden observar a diferentes personas. Algunas de ellas están prácticamente estáticas, realizando escasos movimientos en zonas muy reducidas. Otras aparecen en la escena caminando, bien de forma continuada (sin pararse), o bien realizando paradas de corta duración.

Haciendo uso del algoritmo **FGD**, se obtienen los siguientes resultados:

	NR	MP	RM	WQM
Parámetros por defecto	0,76313854	0,89737637	0,01976978	122,993161
Lc = 32	0,79877772	0,91582011	0,0204747	150,293429
Lc = 128	0,75680161	0,88332687	0,01914288	122,232227
Lcc = 16	0,85498832	0,80166711	0,02136615	163,853226
Lcc = 64	0,77325792	0,868724	0,01852968	126,318487
N1c = 15 N2c = 25	0,76317909	0,89720034	0,01973862	122,328953
N1c =45 N2c = 75	0,76303115	0,8973111	0,01976557	122,840678
N1cc = 25 N2cc = 40	0,78533254	0,90716924	0,02042462	133,817599
N1cc = 75 N2cc = 120	0,76310155	0,89352082	0,01965917	122,218181
alpha1 = 0,3	0,81066965	0,87019138	0,01981211	126,074302
alpha1 = 0,5	0,81857013	0,86048304	0,02004435	132,244442
alpha1 = 0,7	0,83387705	0,82923232	0,02011379	123,105265
alpha1 = 0,9	0,83224522	0,8612126	0,02064327	126,396728
alpha2 = 0,01	0,76411214	0,90696047	0,01999047	124,996935
alpha2 = 0,1	0,819866	0,9259673	0,01637795	105,22139
alpha3 = 0,3	0,75127548	0,90041235	0,02058464	118,557055
alpha3 = 0,7	0,74972063	0,8963814	0,02078659	131,019711
alpha3 = 0,9	0,74972063	0,8963814	0,02078659	131,019711

Es conveniente señalar que para valores promedios mostrados en la tabla anterior referentes a las métricas MP, RM y WQM se han tenido únicamente en cuenta los valores en los frames donde se producía la detección de algún objeto. Así, el valor que se muestra para estas métricas aporta mayor información, pero sin olvidarnos del hecho que no se refleja la cantidad de objetos que se detectan. Por este motivo, y al igual que se hizo con el vídeo anterior, se prefiere buscar los valores para los parámetros que mejores resultados obtienen al aplicar la métrica NR.

En ese sentido, fijándonos en la tabla anterior, se puede observar como aumentando el valor del parámetro *Lc*, es decir, poniéndolo con valor 128, se obtienen unos mejores resultados. Entra la duda de qué ocurriría si se aumentara aún más el valor de dicho parámetro. Por tanto, siguiendo con los valores recomendados para este parámetro, se

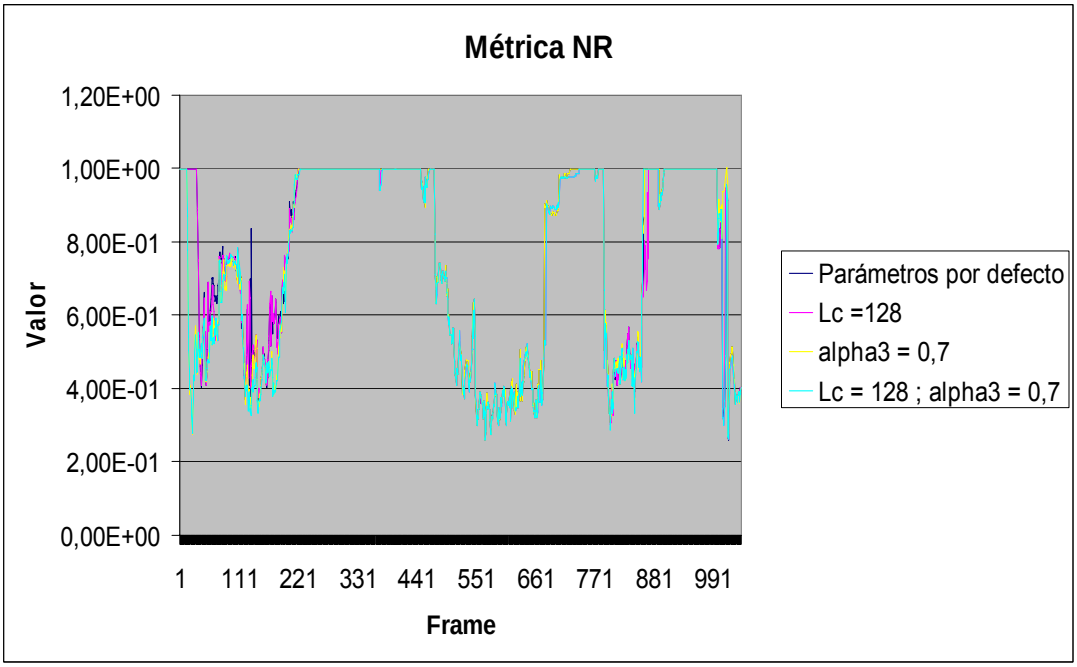
prueba con el siguiente valor mayor, el valor 256. El resultado que se obtiene en este caso para la métrica NR es 0,76389495, por lo que no se mejora el resultado con el anterior valor del parámetro *Lc*.

Por lo tanto, se pasa a evaluar el algoritmo con los valores de los parámetros con los que se ha obtenido mejores resultados para la métrica NR. Los parámetros de esta nueva versión del algoritmo FGD serían los siguientes:

FGD
Lc= 128 *Lcc*= 32 *alpha1*= 0.1 *alpha2*= 0.005 *alpha3*= 0.7 *N1c*= 30 *N2c*= 50 *N1cc*= 50 *N2cc*= 80

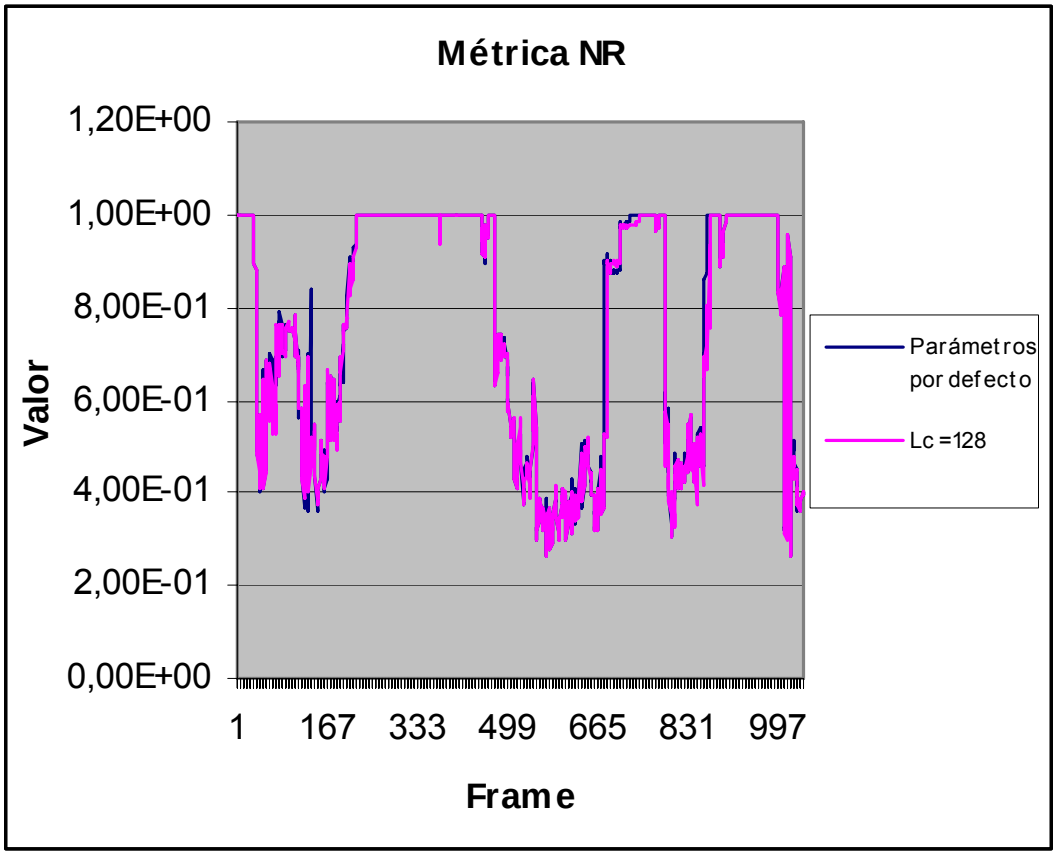
Al aplicar la métrica NR se obtiene como valor 0,74114517, convirtiéndose éste en el mejor de los resultados obtenidos para esta métrica. Se vuelve a confirmar la idea de que mezclando versiones buenas se pueden llegar a obtener resultados mejorados.

A continuación se muestra un gráfico donde se pueden comparar las versiones que mejoran el comportamiento del algoritmo con los parámetros por defecto.



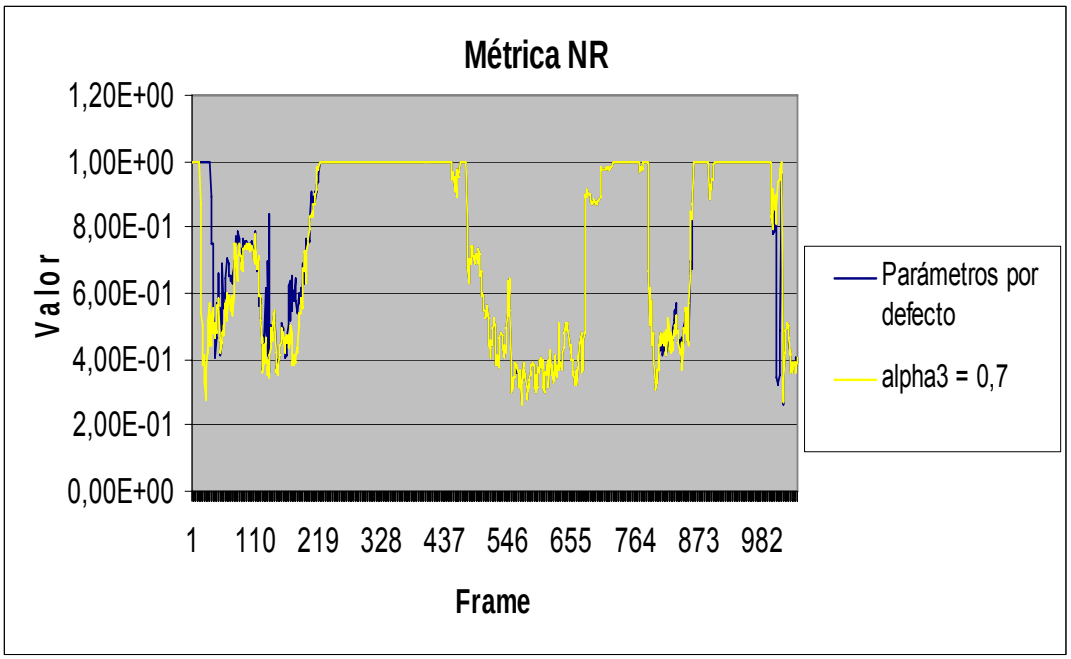
Como se puede apreciar, las diferencias que se observan no son muy significativas, de ahí que el valor promedio de la métrica, aunque se mejore, no varía demasiado. Para poder realizar una mejor comparación, mostraremos gráficos con la versión estándar junto con versiones que la mejoran.

En primero lugar, compararemos la versión estándar con la versión en la que ponemos al parámetro *Lc* el valor 128.



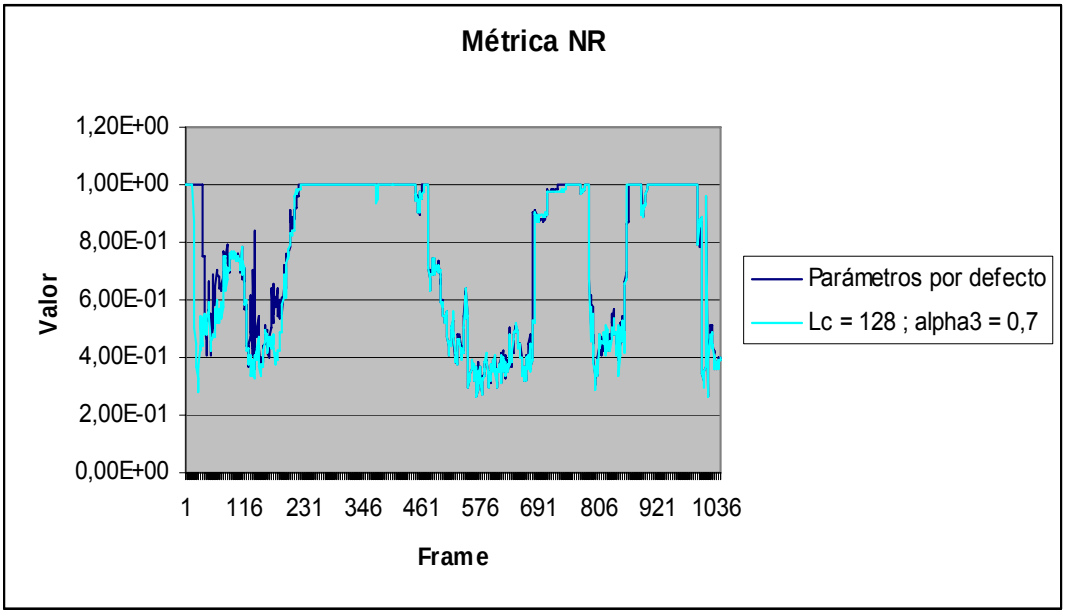
Como se puede apreciar en el gráfico anterior, no se observan grandes diferencias entre ambas versiones. Si se puede notar cómo en ciertos frames se obtienen mejores resultados (valores más bajos) con el parámetro *Lc* puesto a 128.

A continuación, compararemos la versión estándar con la versión del algoritmo con el parámetro *alpha3* puesto a 0,7.



Aquí podemos apreciar la principal mejora que se observó visualmente, que es la detección más temprana cuando se aumenta el valor del parámetro *alpha3*. Con este aumento de dicho parámetro se detectan antes los objetos que aparecen en la escena recién comenzado el vídeo.

Por último, comparamos la versión estándar del algoritmo con la versión con la que se obtienen mejores resultados.



En esta última versión se pueden apreciar las mejoras conjuntas de las vistas en los dos gráficos anteriores.

Haciendo uso del algoritmo **MOG**, se obtienen los siguientes resultados:

	NR	MP	RM	WQM
NG = 3	0,37279273	1,10421258	0,02014225	102,593715
NG = 4	0,39100277	1,10314256	0,01945016	95,6607636
NG = 5	0,39081612	1,1023532	0,01961667	95,6963414

Fijándonos en la tabla anterior y centrándonos en la métrica NR, se puede observar como a raíz que vamos aumentando el valor del parámetro *NG* respecto al valor por defecto, los resultados que se obtienen son peores. Entra la duda de qué ocurriría si se redujera el valor de dicho parámetro. Por tanto, se prueba con el siguiente valor menor, el valor 2. El resultado que se obtiene en este caso para la métrica NR es 0,36371243, por lo que se mejora el resultado obtenido con el anterior valor del parámetro *NG*. Y siguiendo con la misma filosofía, se volvería a probar con el siguiente valor menor para el parámetro NG, el valor 1, para ver si el resultado sigue mejorando. Visualmente, para el valor 1, se observa un comportamiento muy pobre, siendo el resultado que se obtiene para la métrica NR igual a 0,95692538. Por tanto, podemos concluir que es con el parámetro *NG* con valor igual a 2 cuando se obtiene un mejor comportamiento del algoritmo MOG sobre este segundo vídeo, si bien se vuelve a confirmar que las diferencias no son muy significativas.

Finalmente, quedaría volver a comparar los dos algoritmos entre sí sobre este segundo vídeo. Para ello, partiremos de los mejores resultados que se obtienen en cada uno de ellos, y que mostramos en la siguiente tabla:

	NR	MP	RM	WQM
FGD	0,74114517	0,80166711	0,01637795	105,22139
MOG	0,36371243	1,1023532	0,01945016	95,6607636

Como se puede observar en los resultados de la tabla, se prefiere el algoritmo **MOG** si se tiene en cuenta cómo es la detección global a lo largo del vídeo. En este segundo vídeo, si tenemos en cuenta cómo es la detección de los objetos, según la métrica que usemos se preferirá utilizar un algoritmo u otro. Como son tres las métricas que miden esto, y siendo dos las que dan mejores resultados para el algoritmo **FGD**, se prefiere éste en función de este segundo criterio. Este cambio respecto al anterior vídeo se puede justificar por dos motivos. Por un lado, la menor presencia de sombras de los objetos, lo que favorece al algoritmo MOG, y, por otro lado, la pobre detección que se observa, en ocasiones, con el algoritmo FGD sobre algunos objetos, de los que detecta únicamente reducidas regiones de ellos.

Con estos resultados se confirma, por lo tanto, la conclusión que se obtenía de la fase de experimentación, donde se observaba un comportamiento general mejor con el algoritmo **MOG**, ya que, aunque las detecciones de los objetos por separado no fueran tan "perfectas" (por ejemplo, por considerar foreground a las sombras de los objetos), sí que era globalmente mejor teniendo en cuenta que no presentaba tantos problemas a la hora de detectar objetos que se encontraban lejanos a la cámara u objetos que apenas se mueven.

6. CONCLUSIONES

Con la realización de este Proyecto Fin de Carrera se han cumplido los objetivos marcados previamente. Dichos objetivos se resumieron en los siguientes puntos:

- Análisis teórico de los algoritmos FGD y MOG
- Análisis de la implementación de los algoritmos con OpenCV.
- Estudio de las herramientas utilizadas para evaluar los algoritmos.
- Validación de los algoritmos FGD y MOG con distintos Datasets (conjuntos de datos) públicos.

En primer lugar, se realizó una explicación teórica sobre los algoritmos FGD y MOG. Consistió en un estudio detallado, con en el que se llegó a entender en qué se fundamentaban ambos algoritmos para realizar la segmentación.

A continuación, se experimentó con los dos algoritmos sobre la batería de vídeos disponible. Con esto, se llegaron a sacar una serie de diferencias entre ambos, con el objetivo de concluir para qué tipo de vídeos será mejor un algoritmo que otro, y cuáles serán los parámetros más adecuados.

Realizado lo anterior, se vieron diferentes medios que existen para evaluar los algoritmos. Primeramente, se analizó el servicio de evaluación on-line que proporciona **PETS** (**P**erformance **E**valuation of **T**racking and **S**urveillance), donde se evalúan los diferentes algoritmos basándose en cuatro métricas, y los compara con otros con el mismo propósito. Seguidamente, se explicaron las cuatros métricas anteriores destinadas a evaluar los algoritmos.

Y por último, se realizó la evaluación sobre los resultados obtenidos de la experimentación. Para ello, se tuvieron que implementar las cuatro métricas, ya que el servicio on-line de PETS no parecía funcionar correctamente, y se aplicaron éstas sobre los resultados obtenidos de aplicar nuestros algoritmos de segmentación con diferentes valores para sus parámetros. De esta forma, se consigue comparar el comportamiento que tiene cada algoritmo en función de los valores que se den a sus parámetros.

Este último punto hace que nos planteemos también futuros trabajos que se pueden abordar a raíz de todo el trabajo realizado. Concretamente en lo referente a la evaluación de los algoritmos. Teniendo en cuenta que se trata de un trabajo laborioso el encontrar qué valores para cada parámetro son los más adecuados, se piensa en la idea de llegar a construir

un sistema inteligente que lleve a cabo esta función. Se trataría básicamente de un sistema que fuera experimentando con los algoritmos probando con diferentes valores para cada parámetro, y, en función de los valores obtenidos al aplicar las métricas sobre los resultados, el sistema fuera teniendo en cuenta estos resultados para ir fusionando diferentes combinaciones de valores de los parámetros donde se hubieran obtenido mejorías en su comportamiento.

BIBLIOGRAFÍA

- Liyuan Li, Weimin Huang, Irene Yu-Hua Gu, and Qi Tian, "*Statistical Modeling of Complex Backgrounds for Foreground Object Detection*"
- Liyuan Li, Weiming Huang, Irene Y.H. Gu and Qi Tian, "*Foreground Object Detection from Videos Containing Complex Background*"
- Chris Stauffer and W.E.L Grimson, "*Adaptive background mixture models for real-time tracking*"
- Trista P. Chen, Horst Haussecker, Alexander Bovyrin, Roman Belenov, Konstantin Rodyushkin, Alexander Kuranov, and Victor Erujimov "*Computer Vision Workload Analysis: Case Study of Video Surveillance Systems*"
- The OpenCV Video Surveillance / Blob Tracker Facility. (http://opencv.willowgarage.com/wiki/VideoSurveillance)
- Josep Aguilera, Horst Wildenauer, Martin Kampel, Mark Borg, David Thirde, and James Ferryman, "*Evaluation of Motion Segmentation Quality for Aircraft Activity Surveillance* "